LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING

(AUTONOMOUS)

Accredited by NAAC & NBA (Under Tier - I) ISO 9001:2015 Certified Institution

Approved by AICTE, New Delhi. and Affiliated to JNTUK, Kakinada

L.B. REDDY NAGAR, MYLAVARAM, KRISHNA DIST., A.P.-521 230.

http://cse.lbrce.ac.in, cselbreddy@gmail.com, Phone: 08659-222933, Fax: 08659-222931

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BIG DATA WITH HADOOP LAB Manual

(17CI68)

B.Tech. VII SEMESTER – R17

1. Pre-requisites:

- Java Programming
- Database Knowledge

2. <u>Course Educational Objectives:</u>

This course provides practical, foundation level training that enables immediate and effective participation in Big Data and other Analytics projects using Hadoop and R.

3. <u>Course Outcomes:</u>

After the completion of this course, the students will be able to:

CO1: Preparing for data summarization, query, and analysis.

CO2: Applying data modelling techniques to large data sets.

CO3: Creating applications for Big Data analytics.

CO4: Improve individual / teamwork skills, communication & report writing skills with ethical values.

4. Course Articulation Matrix:

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	**3**	**3**	**3**	**2**	**3**	-	-	-	-	-	-	-	**2**	**3**	-
CO2	**3**	**2**	**2**	**2**	**3**	-	-	-	-	-	-	-	**2**	**2**	-
CO3	**3**	**3**	**3**	**2**	**3**	**1**	-	-	-	-	-	-	**2**	**3**	-
CO4	-	-	-	-	-	-	-	**2**	**2**	**2**	-	-	-	-	-
1 - Low **2** –Medium **3** - High															

EXERCISES

Week-1:

Downloading and installing Hadoop; Understanding different Hadoop modes. Start-up scripts, Configuration files.

Week-2:

Hadoop Implementation of file management tasks, such as Adding files and directories, Retrieving files and Deleting files.

Week-3:

Implementation of Matrix Multiplication with Hadoop Map Reduce.

Week-4

Implementation of Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm.

Week-5:

Implementation of K-means clustering using map reduce.

Week-6:

Installation of Hive along with practice examples.

Week-7:

Installation of HBase, Installing thrift along with Practice examples.

Week-8:

Installation of R, along with Practice examples in R

Week-1: Hadoop Configuration

1. (i)Perform setting up and Installing Hadoop in its three operating modes:

- **Standalone**
- **Pseudo distributed**
- **Fully distributed**

(ii) Use web based tools to monitor your Hadoop setup.

Hadoop can run on three modes

a) Standalone mode
b) Pseudo mode
c) Fully distributed mode

The software requirements for Hadoop installation are

- Java Development Kit
- Hadoop framework
- Secured shell

A) STANDALONE MODE:

- Installation of jdk7

  Command: sudo apt-get install openjdk-7-jdk

- Download and extract Hadoop

  Command: wget http://archive.apache.org/dist/hadoop/core/hadoop-1.2.0/hadoop-1.2.0.tar.gz

  Command: tar -xvf hadoop-1.2.0.tar.gz

  Command: sudo mv hadoop-1.2.0 /usr/lib/hadoop

- Set the path for java and hadoop

  Command: sudo gedit $HOME/.bashrc

  ```
  export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386
  export PATH=$PATH:$JAVA_HOME/bin

  export HADOOP_COMMON_HOME=/usr/lib/hadoop
  export HADOOP_MAPRED_HOME=/usr/lib/hadoop
  export PATH=$PATH:$HADOOP_COMMON_HOME/bin
  export PATH=$PATH:$HADOOP_COMMON_HOME/Sbin
  ```

- Checking of java and hadoop

  Command: java -version

  Command: hadoop version

B) PSEUDOMODE:

Hadoop single node cluster runs on single machine. The name nodes and data nodes are performing on the one machine. The installation and configuration steps as given below:

- Installation of secured shell

  Command: sudo apt-get install open ssh-server

- Create a ssh key for password less ssh configuration
  Command: ssh-keygen -t rsa –P ""

- Moving the key to authorized key
  Command: cat $HOME/.ssh/id_rsa.pub >>$HOME/.ssh/authorized_keys

  /**********RESTART THECOMPUTER**********/

- Checking of secured shell login
  Command: ssh localhost

- Add JAVA_HOME directory in hadoop-env.shfile
  Command: sudogedit /usr/lib/hadoop/conf/hadoop-env.sh
  export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386

- Creating name node and data node directories for hadoop **Command:** sudo mkdir-p /usr/lib/hadoop/dfs/name node**Command:** sudomkdir-p/usr/lib/hadoop/dfs/data node

- Configurecore-site.xml
  Command: sudogedit /usr/lib/hadoop/conf/core-site.xml

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:8020</value>
</property>
```

- Configurehdfs-site.xml
  Command: sudogedit /usr/lib/hadoop/conf/hdfs-site.xml

```
<property>
    <name>dfs.replication</name>
    <value>1</value>
</property>
<property>
    <name>dfs.permissions</name>
    <value>false</value>
</property>
<property>
    <name>dfs.name.dir</name>
    <value>/usr/lib/hadoop/dfs/name node</value>
</property>
<property>
    <name>dfs.data.dir</name>
    <value>/usr/lib/hadoop/dfs/data node</value>
</property>
```

- Configuremapred-site.xml

  Command: sudogedit /usr/lib/hadoop/conf/mapred-site.xml

  ```
  <property>
      <name>mapred.job.tracker</name>
      <value>localhost:8021</value>
  </property>
  ```

- Format the name node

  Command: hadoopname node -format

- Start the name node,data node

  Command: start-dfs.sh

- Start the task tracker and jobtracker

  Command: start-mapred.sh

- To check if Hadoop startedcorrectly

  Command: jps

  name node
  secondarynamenode
  data node
  jobtracker
  tasktracker

C) FULLY DISTRIBUTEDMODE:

All the demons like name nodes and data nodes are runs on different machines. The data will replicate according to the replication factor in client machines. The secondary name node will store the mirror images of name node periodically. The name node having the metadata where the blocks are stored and number of replicas in the client machines. The slaves and master communicate each other periodically. The configurations of multinode cluster are given below:

- Configure the hosts in allnodes/machines

  Command: sudogedit /etc/hosts/

  ```
  192.168.1.58   pcetcse1
  192.168.1.4    pcetcse2
  192.168.1.5    pcetcse3
  192.168.1.7    pcetcse4
  192.168.1.8    pcetcse5
  ```

- PasswordlessSshConfiguration

  - Create ssh key onname node/master.
    Command: ssh-keygen -t rsa -p “”

  - Copy the generated public key alldata nodes/slaves.

Command: ssh-copy-id -i~/.ssh/id_rsa.pubhuser@pcetcse2
Command: ssh-copy-id -i~/.ssh/id_rsa.pubhuser@pcetcse3
Command: ssh-copy-id -i~/.ssh/id_rsa.pubhuser@pcetcse4
Command: ssh-copy-id -i~/.ssh/id_rsa.pubhuser@pcetcse5

/**********RESTART ALL NODES/COMPUTERS/MACHINES **********/

NOTE: Verify the passwordlessssh environment from name node to all data nodes as "hduser" user.

- Login to master node
  Command: sshpcetcse1
  Command: sshpcetcse2
  Command: sshpcetcse3
  Command: sshpcetcse4
  Command: sshpcetcse5

- Add JAVA_HOME directory in hadoop-env.sh file in allnodes/machines
  Command: sudogedit /usr/lib/hadoop/conf/hadoop-env.sh
  export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386

- Creating name node directory inname node/master
  Command: sudomkdir -p /usr/lib/hadoop/dfs/name node

- Creating name node directory indatanonodes/slaves
  Command: sudomkdir -p /usr/lib/hadoop/dfs/data node

- Configure core-site.xml in allnodes/machines
  Command: sudogedit /usr/lib/hadoop/conf/core-site.xml

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://pcetcse1:8020</value>
</property>
```

- Configure hdfs-site.xml inname node/master
  Command: sudogedit /usr/lib/hadoop/conf/hdfs-site.xml

```
<property>
    <name>dfs.replication</name>
    <value>3</value>
</property>
<property>
    <name>dfs.permissions</name>
    <value>false</value>
</property>
<property>
    <name>dfs.name.dir</name>
    <value>/usr/lib/hadoop/dfs/name node</value></property>
```

- Configure hdfs-site.xml in data nodes/slaves

  Command: sudogedit /usr/lib/hadoop/conf/hdfs-site.xml

```
<property>
        <name>dfs.replication</name>
        <value>3</value>
</property>
<property>
        <name>dfs.permissions</name>
        <value>false</value>
</property>
<property>
        <name>dfs.data.dir</name>
        <value>/usr/lib/hadoop/dfs/data node</value>
</property>
```

- Configure mapred-site.xml in allnodes/machines

  Command: sudogedit /usr/lib/hadoop/conf/mapred-site.xml

```
<property>
        <name>mapred.job.tracker</name>
        <value>pcetcse1:8021</value>
</property>
```

- Configure masters in all name node/master give the secondary name nodehostname

  Command: sudogedit /usr/lib/hadoop/conf/masters

```
pcetcse2
```

- Configure masters in all data nodes/slaves give the name nodehostname

  Command: sudogedit /usr/lib/hadoop/conf/masters

```
pcetcse1
```

- Configure slaves in allnodes/machines

  Command: sudogedit/usr/lib/hadoop/conf/slaves

```
pcetcse2
pcetcse3
pcetcse4
pcetcse5
```

- Format the name node

  Command: hadoopname node -format

- Start the name node,data node

  Command: start-dfs.sh

- Start the task tracker and jobtracker

  Command: start-mapred.sh

- To check if Hadoop started correctly check in all thenodes/machines

huser@pcetcse1:$jps
name
nodejobtr
acker

huser@pcetcse2:$jps
secondaryname
nodetasktrackerdata
node

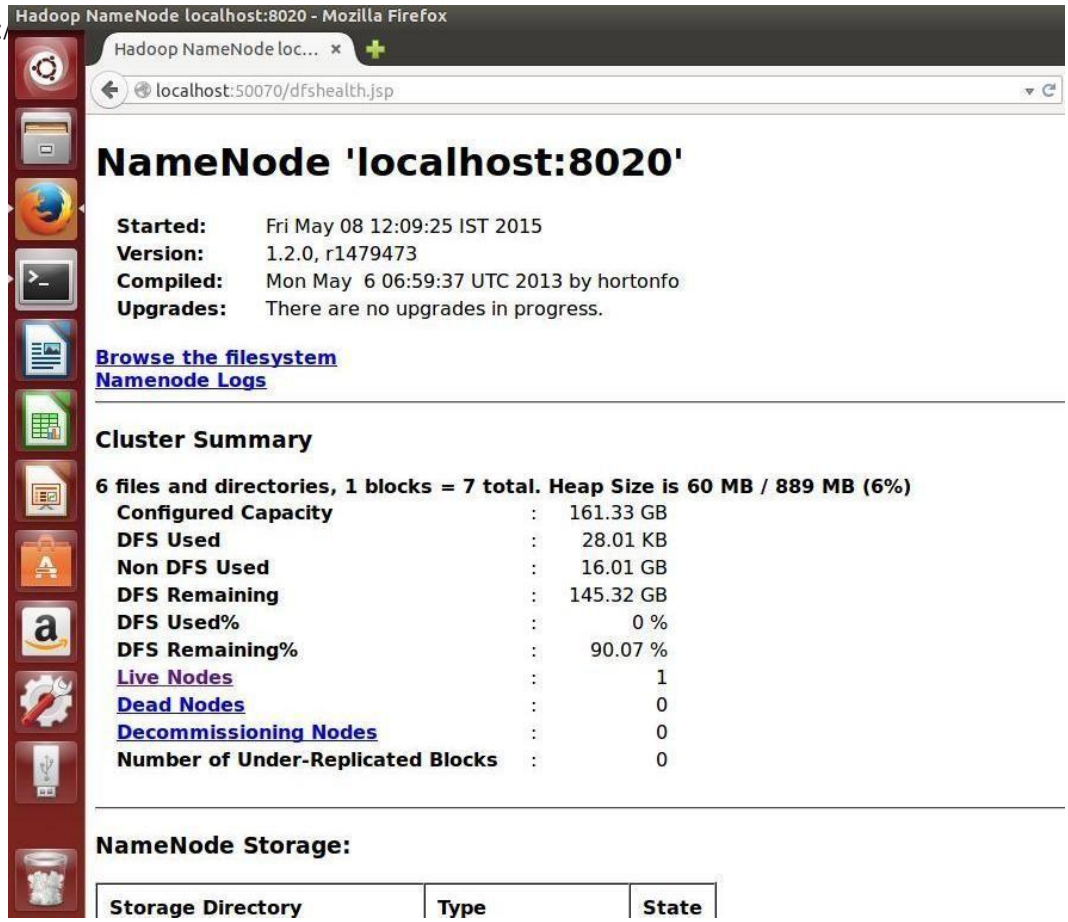
huser@pcetcse3:$jps
data
nodetasktracke
r

huser@pcetcse4:$jps
data
nodetasktracke
r

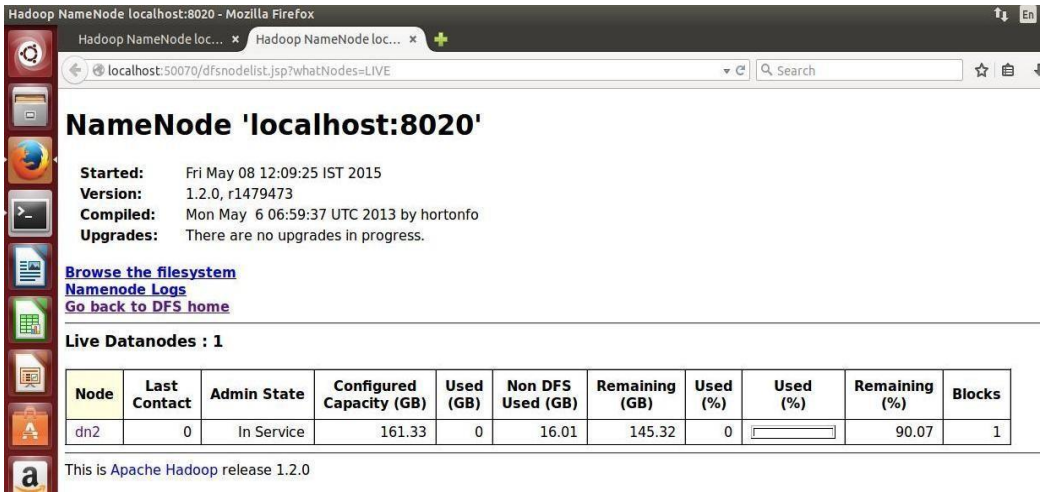
huser@pcetcse5:$jps
data
nodetasktracke
r

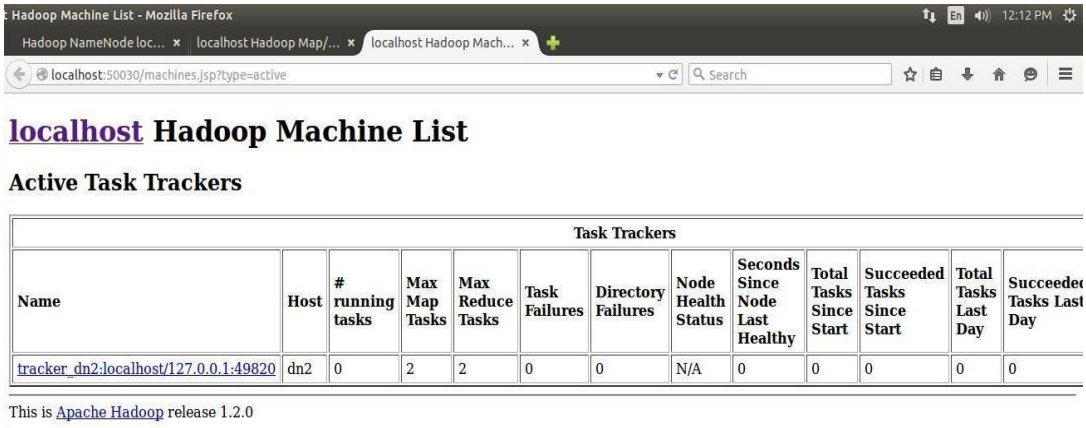
Using HDFS monitoring UI

- HDFS Name node on UI http:/

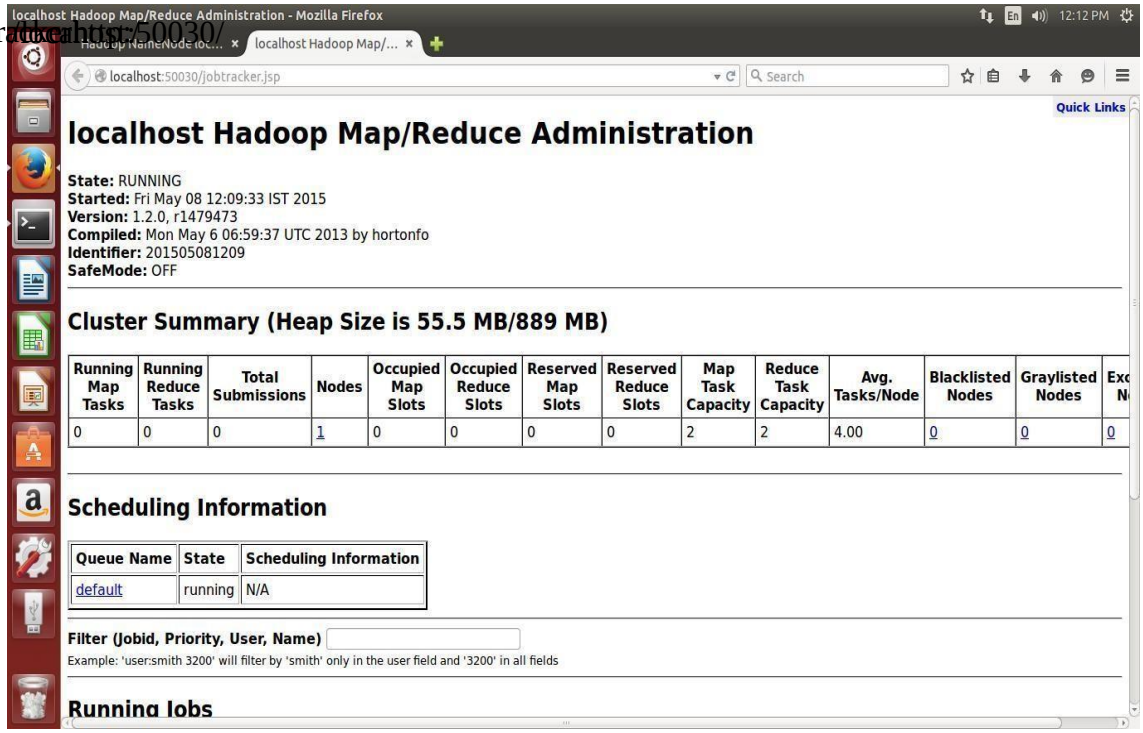


- HDFS Live Nodeslist

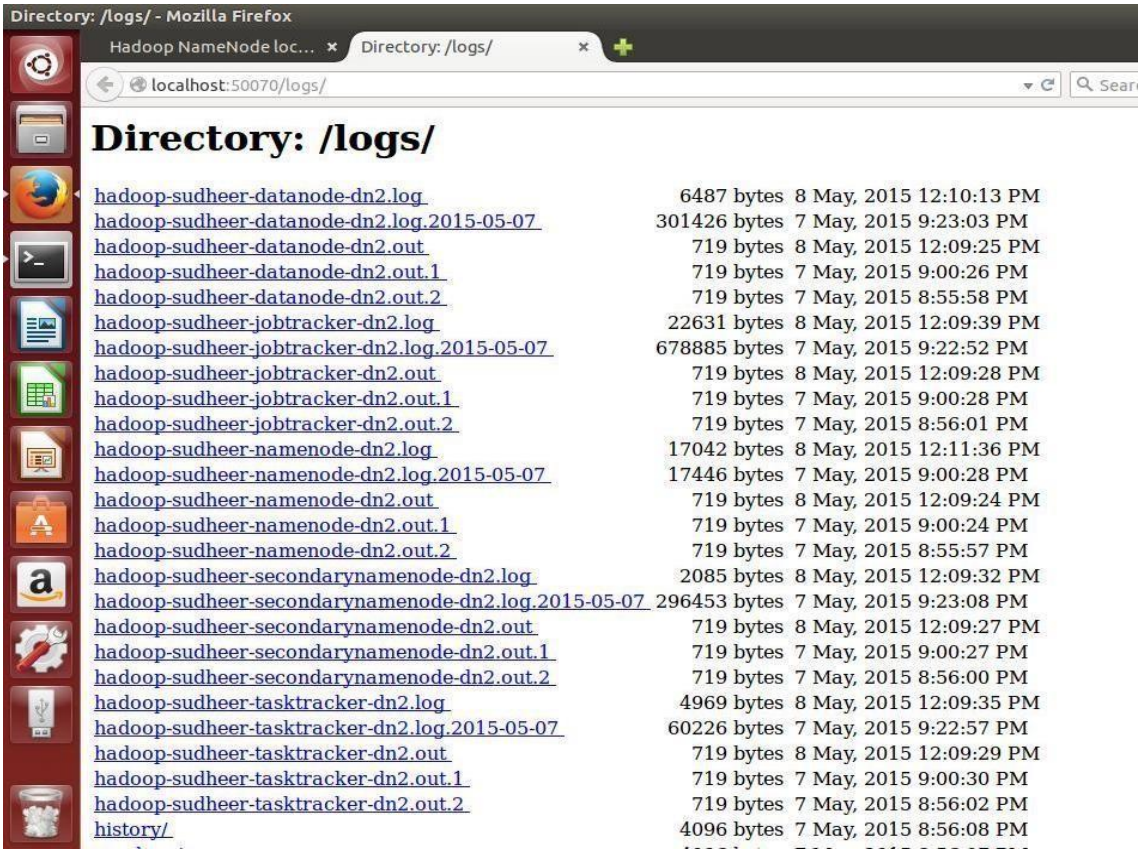




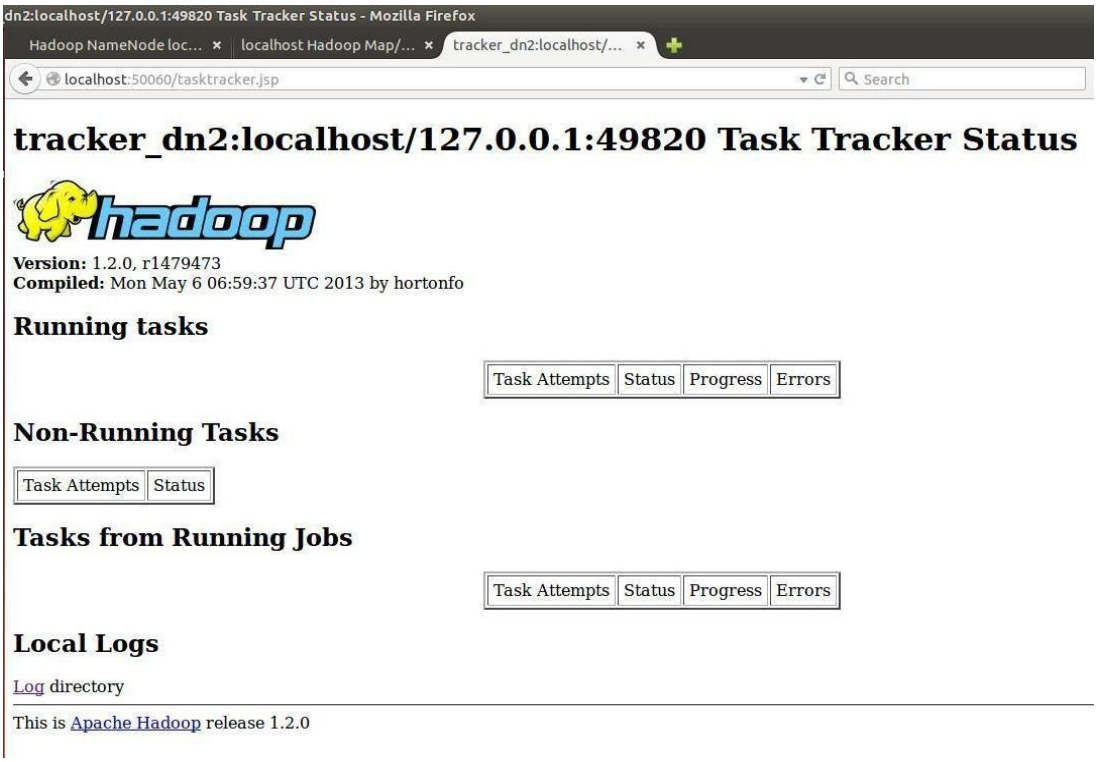
- HDFS Jobtracker: http://localhost:50030/



- HDFS Logs
  http://locahost:50070/logs/



- HDFS Tasktracker
  http://locahost:50060/



Week -2: HDFS

2. Implement the following file management tasks in Hadoop:

- **Adding files and directories**
- **Retrieving files**
- **Deleting files**

Hint: A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.

HDFS basic Command-line file operations

1. Create a directory in HDFS at givenpath(s):
Command: hadoop fs -mkdir<paths>

2. List the contents of adirectory:
Command: hadoop fs -ls <args>

3. Upload and download a file inHDFS:
Upload:
Command: hadoop fs -put <localsrc><HDFS_dest_path>
Download:
Command: hadoop fs -get <HDFS_src><localdst>

4. See contents of afile:
Command: hadoop fs -cat <path[filename]>

5. Copy a file from source todestination:
Command: hadoop fs -cp <source><dest>

6. Copy a file from/To Local file system toHDFS:
Command: hadoop fs -copyFromLocal<localsrc> URI
Command: hadoop fs -copyToLocal [-ignorecrc] [-crc] URI <localsrc>

7. Move file from source todestination:
Command: hadoop fs -mv <src><dest>

8. Remove a file or directory inHDFS:
Remove files specified as argument. Delete directory only when it is empty.
Command: hadoop fs -rm <arg>
Recursive version of delete
Command: hadoop fs -rmr<arg>

9. Display last few lines of afile:
Command: hadoop fs -tail <path[filename]>

10. Display the aggregate length of afile:
Command: hadoop fs -du <path>

11. Gettinghelp:
Command: hadoop fs -help

Adding files and directories:

- Creating adirectory
  Command: hadoop fs -mkdir input/
- Copying the files from localfile system toHDFS
  Command: hadoopfs -put inp/file01input/

Retrieving files:

Command: hadoop fs -get input/file01 localfs

Deleting files and directories:

Command: hadoop fs -rmr input/file01

Week -3: HDFS - Map Reduce

3) Implement Matrix Multiplication with Hadoop MapReduce.

PROGRAM:

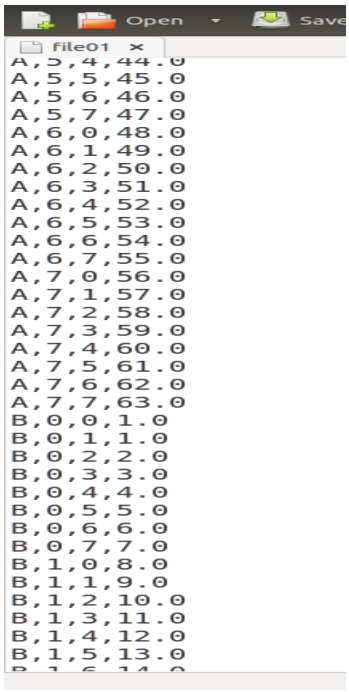
```
import java.io.IOException;
import java.util.*;
 import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
public class MatrixMul {
        /**********Mapper class**********/
public static class Map extends Mapper<LongWritable, Text, Text, Text> {
public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
        Configuration conf = context.getConfiguration();
        int m = Integer.parseInt(conf.get("m"));
        int p = Integer.parseInt(conf.get("p"));
        String line = value.toString();
        String[] indicesAndValue = line.split(",");
        Text outputKey = new Text();
        Text outputValue = new Text();
        if (indicesAndValue[0].equals("A")) {
           for (int k = 0; k < p; k++)
           {
                outputKey.set(indicesAndValue[1] + "," + k);
                outputValue.set("A," + indicesAndValue[2] + "," + indicesAndValue[3]);
                context.write(outputKey, outputValue);
           }
        } else {
           for (int i = 0; i< m; i++) {
                outputKey.set(i + "," + indicesAndValue[2]);
                outputValue.set("B," + indicesAndValue[1] + "," + indicesAndValue[3]);
                context.write(outputKey, outputValue);
            }
        }
    }
}
```

```
 /*************************Reducer Class************************************/
public static class Reduce extends Reducer<Text, Text, Text, Text> {
public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
        String[] value;
        HashMap<Integer, Float>hashA = new HashMap<Integer, Float>();
        HashMap<Integer, Float>hashB = new HashMap<Integer, Float>();
        for (Text val : values) {
                value = val.toString().split(",");
                if (value[0].equals("A")) {
                hashA.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));
                } else {
                hashB.put(Integer.parseInt(value[1]), Float.parseFloat(value[2]));
                }
        }
        double[] myList = new double[10];
        int n = Integer.parseInt(context.getConfiguration().get("n"));
        float result = 0.0f;
        float a_ij;
        float b_jk;
        for (int j = 0; j < n; j++) {
                a_ij = hashA.containsKey(j) ?hashA.get(j) : 0.0f;
                b`_jk = hashB.containsKey(j) ? hashB.get(j) : 0.0f;
                result += a_ij * b_jk;
        }
        if (result != 0.0f) {
           context.write(null, new Text(key.toString() + "," + Float.toString(result)));
        }
      }
  }

   /*********************Driver(main) function*********************************/
public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        // A is an m-by-n matrix; B is an n-by-p matrix.
        conf.set("m", "8");
        conf.set("n","8");
        conf.set("p","8");

        Job job = Job.getInstance(conf, "MatrixMultiplication");
        job.setJarByClass(MatrixMul.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
```

```
        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.submit();
}
}
```

- Create the temporary content file in the inputdirectory
  Command: sudomkdir input
  Command: sudogedit input/matrix.txt
- enter the 8x8 matrix on that file

Sample matrix 8x8 matrixdataset



- Put the matrix input intoHDFS
  Command: hadoop fs -mkdirinputMatrix
  Command: hadoop fs -put input/matrix.txt inputMatrix/
- Create jar file MatrixMultiplicationProgram
  Command: hadoopcom.sun.tools.javac.Main MatrixMul.java
  Command: jar cvf mc.jar MatrixMul *.class
- Run mc jar file on inputdirectory
  Command: hadoop jar mc.jar MatrixMulinputMatrix/matrix.txt out1

- To see the output browse the filesystem

Week -4: HDFS - Map Reduce

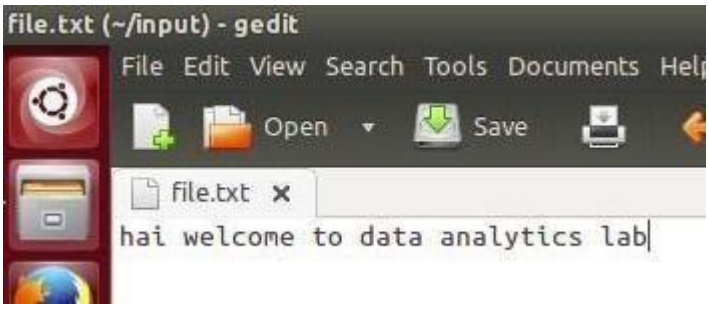
4) Implementation of Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm.

PROGRAM:

```
import java.io.IOException;
import java.util.StringTokenizer;
importorg.apache.hadoop.conf.Configuration;
importorg.apache.hadoop.fs.Path;
importorg.apache.hadoop.io.IntWritable;
importorg.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount {
 public static class TokenizerMapper
 extends Mapper<Object, Text, Text, IntWritable>{
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();
   public void map(Object key, Text value, Context context)
         throws IOException, InterruptedException {
    StringTokenizeritr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
     word.set(itr.nextToken());
     context.write(word, one);
    }
   }
  }
public static class IntSumReducer
    extends Reducer<Text,IntWritable,Text,IntWritable> {
 private IntWritable result = new IntWritable();
 public void reduce(Text key, Iterable<IntWritable> values, Context context
         ) throws IOException, InterruptedException{
    int sum = 0;
    for (IntWritableval : values) {
         sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
   }}
```

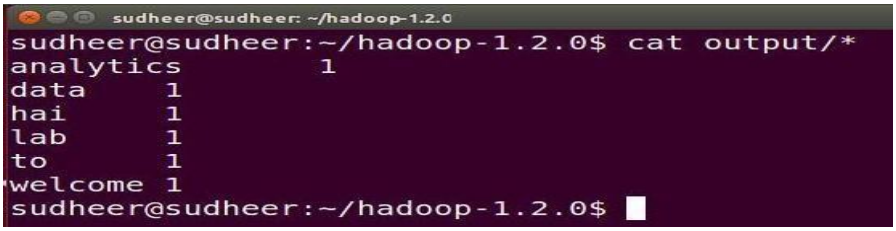
```
  public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
      }
}
```

- Create the temporary content file in the inputdirectory
  Command: sudomkdir input
  Command: sudogedit input/file.txt
- Type some text on that file, save the file andclose



- Put the file.txt intohdfs
  Command: hadoop fs -mkdir input
  Command: hadoop fs -put input/file.txt input/
- Create jar file WordCountProgram
  Command: hadoopcom.sun.tools.javac.Main WordCount.java
  Command: jar cf wc.jar WordCount*.class
- Run WordCountjar file on inputdirectory
  Command: hadoop jar wc.jar WordCount input output

- To see theoutput
  Command: cat output/*



Week -5: HDFS - Map Reduce - Clustering

5) Implementation of K-means clustering using map reduce.

```
packagecom.lbrce.kmean;
importjava.io.IOException;
importjava.util.*;
importjava.io.*;
importorg.apache.hadoop.conf.Configuration;
importorg.apache.hadoop.filecache.DistributedCache;
importorg.apache.hadoop.fs.FileSystem;
importorg.apache.hadoop.fs.Path;
importorg.apache.hadoop.io.*;
importorg.apache.hadoop.mapred.*;
importorg.apache.hadoop.mapred.Reducer;
@SuppressWarnings("deprecation")
publicclassKMean {
publicstatic String OUT = "outfile";
publicstatic String IN = "inputlarger";
publicstatic String CENTROID_FILE_NAME = "/centroid.txt";
publicstatic String OUTPUT_FILE_NAME = "/part-00000";
publicstatic String DATA_FILE_NAME = "/data.txt";
publicstatic String JOB_NAME = "KMeans";
publicstatic String SPLITTER = "t| ";
publicstatic List<Double>mCenters = newArrayList<Double>();
publicstaticclass Map extendsMapReduceBaseimplements Mapper<LongWritable, Text,
DoubleWritable, DoubleWritable> {
    @Override
publicvoidconfigure(JobConf job) {
try {
Path[] cacheFiles = DistributedCache.getLocalCacheFiles(job);
if (cacheFiles != null&&cacheFiles.length> 0) {
            String line;
mCenters.clear();
BufferedReadercacheReader = newBufferedReader(
newFileReader(cacheFiles[0].toString()));
try {
while ((line = cacheReader.readLine()) != null) {
String[] temp = line.split(SPLITTER);
mCenters.add(Double.parseDouble(temp[0]));
              }
            } finally {
cacheReader.close();
            }
          }
        } catch (IOException e) {
System.err.println("Exception reading DistribtuedCache: " + e);
        }
    }
    @Override
publicvoidmap(LongWritable key, Text value, OutputCollector<DoubleWritable, DoubleWritable>
output, Reporter reporter) throwsIOException {
        String line = value.toString();
double point = Double.parseDouble(line);
double min1, min2 = Double.MAX_VALUE, nearest_center = mCenters.get(0);
for (doublec :mCenters) {
```

```
            min1 = c - point;
        if (Math.abs(min1) <Math.abs(min2)) {
        nearest_center = c;
                    min2 = min1;
                }
            }
        output.collect(newDoubleWritable(nearest_center),
        newDoubleWritable(point));
          }
       }
        publicstaticclass Reduce extendsMapReduceBaseimplements
            Reducer<DoubleWritable, DoubleWritable, DoubleWritable, Text> {
            @Override
        publicvoidreduce(DoubleWritable key, Iterator<DoubleWritable> values,
        OutputCollector<DoubleWritable, Text> output, Reporter reporter)throwsIOException {
        doublenewCenter;
        double sum = 0;
        intno_elements = 0;
            String points = "";
        while (values.hasNext()) {
        double d = values.next().get();
                points = points + " " + Double.toString(d);
                sum = sum + d;
                ++no_elements;
            }
        newCenter = sum / no_elements;
        output.collect(newDoubleWritable(newCenter), new Text(points));
          }
       }
        publicstaticvoidmain(String[] args) throws Exception {
        run(args);
       }
        publicstaticvoidrun(String[] args) throws Exception {
        IN = args[0];
        OUT = args[1];
            String input = IN;
            String output = OUT + System.nanoTime();
            String again_input = output;
        int iteration = 0;
        booleanisdone = false;
        while (isdone == false) {
        JobConf conf = newJobConf(KMean.class);
        if (iteration == 0) {
                Path hdfsPath = newPath(input + CENTROID_FILE_NAME);
        DistributedCache.addCacheFile(hdfsPath.toUri(), conf);
            } else {
                Path hdfsPath = newPath(again_input + OUTPUT_FILE_NAME);
        DistributedCache.addCacheFile(hdfsPath.toUri(), conf);
            }
        conf.setJobName(JOB_NAME);
        conf.setMapOutputKeyClass(DoubleWritable.class);
        conf.setMapOutputValueClass(DoubleWritable.class);
        conf.setOutputKeyClass(DoubleWritable.class);
        conf.setOutputValueClass(Text.class);
        conf.setMapperClass(Map.class);
        conf.setReducerClass(Reduce.class);
        conf.setInputFormat(TextInputFormat.class);
```

```
conf.setOutputFormat(TextOutputFormat.class);
FileInputFormat.setInputPaths(conf, newPath(input + DATA_FILE_NAME));
FileOutputFormat.setOutputPath(conf, new Path(output));
JobClient.runJob(conf);
        Path ofile = newPath(output + OUTPUT_FILE_NAME);
FileSystem fs = FileSystem.get(new Configuration());
BufferedReaderbr = newBufferedReader(newInputStreamReader(fs.open(ofile)));
        List<Double>centers_next = newArrayList<Double>();
        String line = br.readLine();
while (line != null) {
String[] sp = line.split("t| ");
double c = Double.parseDouble(sp[0]);
centers_next.add(c);
           line = br.readLine();
        }
br.close();
        String prev;
if (iteration == 0) {
prev = input + CENTROID_FILE_NAME;
        } else {
prev = again_input + OUTPUT_FILE_NAME;
        }
        Path prevfile = new Path(prev);
FileSystem fs1 = FileSystem.get(new Configuration());
BufferedReader br1 = newBufferedReader(newInputStreamReader(fs1.open(prevfile)));
        List<Double>centers_prev = newArrayList<Double>();
        String l = br1.readLine();
while (l != null) {
String[] sp1 = l.split(SPLITTER);
double d = Double.parseDouble(sp1[0]);
centers_prev.add(d);
           l = br1.readLine();
        }
        br1.close();
Collections.sort(centers_next);
Collections.sort(centers_prev);

        Iterator<Double> it = centers_prev.iterator();
for (doubled :centers_next) {
double temp = it.next();
if (Math.abs(temp - d) <= 0.1) {
isdone = true;
           } else {
isdone = false;
break;
           }
        }
        ++iteration;
again_input = output;
        output = OUT + System.nanoTime();
     }
   }
}
```

Step 01: Copy the 17761A05A3KMeans.jar jar file from Local Machine to Student server using WinSCP.

Step 02: Copying the jar file from Student server to individual directory (i.e. 17761A05A3) present in hduser server.

```
student@BIGDATA:~$ scp 17761A05A3KMeans.jar hduser@192.168.100.20:/home/hduser/17761A05A3
hduser@192.168.100.20's password:
17761A05A3KMeans.jar                                  100% 6598     4.9MB/s   00:00
```

Step 03: Directing to hduserso as to perform operations on the jar file.

```
student@BIGDATA:~$ ssh hduser@192.168.100.20
hduser@192.168.100.20's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.4.0-62-generic x86_64)
```

Step 04: Checking whether our directory is present.

```
hduser@master-node-20:~$ ls
 17761A0551      file1                  Pictures
 17761A0560      gow                    pig_1602664981900.log
 17761A0570      hari                   pig_1605760283597.log
 17761A0583      hbase                  pig_1608007059076.log
 17761A0584      HBASE                  pig_1611203839844.log
 17761A0597      HdfsRead.class         pig_1611206548210.log
 17761A0598      HdfsWrite.class        pig_1611638008108.log
 17761A05A3      hduser@192.168.100.20  pig_1611638245577.log
```

Step 05: Changing the path from hduser to our directory.

```
hduser@master-node-20:~$ cd 17761A05A3
hduser@master-node-20:~/17761A05A3$
```

Step 06: Checking whether the jar file was successfully copied or not.

```
hduser@master-node-20:~/17761A05A3$ ls
17761A05A3charcount.jar  17761A05A3Matrix.jar     centroid.txt  example
17761A05A3KMeans.jar     17761A05A3wordcount.jar  data.txt
```

Step 07: Now for performing KMeans operation we require two input files which are text files (i.e. named as centroid.txt and data.txt). The centroid.txt consists of centroids and, data.txt consists of data points.

```
hduser@master-node-20:~/17761A05A3$ cat centroid.txt
20.00
30.00
40.00hduser@master-node-20:~/17761A05A3$ cat data.txt
20
23
19
29
43
35
18
25
27
```

Step 08: As the input files are created in the local file system, we have to copy this file from the local file system to Hadoop file system in which the actual operation takes place.

```
hduser@master-node-20:~/17761A05A3$ hadoop fs -put centroid.txt /17761A05A3
hduser@master-node-20:~/17761A05A3$ hadoop fs -put data.txt /17761A05A3
hduser@master-node-20:~/17761A05A3$ hadoop fs -ls /17761A05A3
Found 9 items
drwxr-xr-x   - hduser supergroup          0 2021-01-28 14:55 /17761A05A3/885142762271507
drwxr-xr-x   - hduser supergroup          0 2021-01-28 14:59 /17761A05A3/885539032366797
-rw-r--r--   2 hduser supergroup         64 2021-01-28 13:47 /17761A05A3/MN
drwxr-xr-x   - hduser supergroup          0 2021-01-28 13:59 /17761A05A3/MatOutput
-rw-r--r--   2 hduser supergroup         17 2021-01-28 15:10 /17761A05A3/centroid.txt
drwxr-xr-x   - hduser supergroup          0 2021-01-28 13:19 /17761A05A3/charcount
-rw-r--r--   2 hduser supergroup         27 2021-01-28 15:10 /17761A05A3/data.txt
-rw-r--r--   2 hduser supergroup         58 2021-01-28 12:23 /17761A05A3/example
drwxr-xr-x   - hduser supergroup          0 2021-01-28 12:53 /17761A05A3/wordcount
```

Step 09: After successfully copying the input file to the Hadoop Filesystem now we have to execute the jar file. Additionally, the output directory must becreated.

```
hduser@master-node-20:~/17761A05A3$ hadoop jar 17761A05A3KMeans.jar /17761A05A3/ /17761A0
5A3/
```

Step 10: Checking whether the job is done successfully or not and open the output directory.

```
21/01/28 12:53:24 INFO mapreduce.Job:  map 100% reduce 100%
21/01/28 12:53:24 INFO mapreduce.Job: Job job_1611740689619_0058 completed successfully
```

```
hduser@master-node-20:~/17761A05A3$ hadoop fs -ls /17761A05A3/885539032366797
Found 2 items
-rw-r--r--   2 hduser supergroup          0 2021-01-28 14:59 /17761A05A3/885539032366797/
_SUCCESS
-rw-r--r--   2 hduser supergroup         77 2021-01-28 14:59 /17761A05A3/885539032366797/
part-00000
```

Step 11: Now opening the output file in read mode to print the output of the operation.

```
hduser@master-node-20:~/17761A05A3$ hadoop fs -cat /17761A05A3/885539032366797/part-00000

21.0     25.0 18.0 19.0 23.0 20.0
30.333333333333332       29.0 27.0 35.0
43.0     43.0
```

The output after execution of jar file will be displayed.

Week – 6: HIVE

6) Installation of Hive along with practice examples.

- **Download and extract Hive:**

Command: wgethttps://archive.apache.org/dist/hive/hive-0.14.0/apache-hive-0.14.0-bin.tar.gz

Command: tar zxvf apache-hive-0.14.0-bin.tar.gz

Command: sudo mv apache-hive-0.13.1-bin /usr/lib/hive

Command: sudogedit $HOME/.bashrc

```
export HIVE_HOME=/usr/lib/hive
export PATH=$PATH:$HIVE_HOME/bin
export CLASSPATH=$CLASSPATH:/usr/lib/hadoop/lib/*.jar
export CLASSPATH=$CLASSPATH:/usr/lib/hive/lib/*.jar
```

Command: sudo cd $HIVE_HOME/conf

Command: sudo cp hive-env.sh.template hive-env.sh

```
export HADOOP_HOME=/usr/lib/hadoop
```

- **Downloading ApacheDerby**

The following command is used to download Apache Derby. It takes some time to download.

Command: wget http://archive.apache.org/dist/db/derby/db-derby-10.4.2.0/db-derby-10.4.2.0-bin.tar.gz

Command: tar zxvf db-derby-10.4.2.0-bin.tar.gz

Command: sudo mv db-derby-10.4.2.0-bin /usr/lib/derby

Command: sudogedit $HOME/.bashrc

```
export DERBY_HOME=/usr/local/derby
export PATH=$PATH:$DERBY_HOME/bin
export
CLASSPATH=$CLASSPATH:$DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/
derbytools.jar:$DERBY_HOME/lib/derbyclient.jar
```

Command: sudomkdir $DERBY_HOME/data

Command: sudo cd $HIVE_HOME/conf

Command: sudo cp hive-default.xml.template hive-site.xml

Command: Sudogedit $HOVE_HOME/conf/hive-site.xml

```
<property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:derby://localhost:1527/metastore_db;create=true</value>
    <description>JDBC connect string for a JDBC metastore</description>
</property>
```

- Create a file named jpox.properties and add the following lines intoit:

```
javax.jdo.PersistenceManagerFactoryClass =
org.jpox.PersistenceManagerFactoryImplorg.jpox.autoCreateSchema = false
org.jpox.validateTables = false
org.jpox.validateColumns = false
org.jpox.validateConstraints = false
org.jpox.storeManagerType =
rdbmsorg.jpox.autoCreateSchema = true
org.jpox.autoStartMechanismMode = checked
org.jpox.transactionIsolation =
read_committedjavax.jdo.option.DetachAllOnCo
mmit = true
javax.jdo.option.NontransactionalRead = true
javax.jdo.option.ConnectionDriverName =
org.apache.derby.jdbc.ClientDriverjavax.jdo.option.ConnectionURL =
jdbc:derby://hadoop1:1527/metastore_db;create = true
javax.jdo.option.ConnectionUserName = APP
javax.jdo.option.ConnectionPassword = mine
```

Command: HADOOP_HOME/bin/hadoop fs -mkdir /tmp
Command: HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse
Command: HADOOP_HOME/bin/hadoop fs -chmodg+w /tmp
Command: HADOOP_HOME/bin/hadoop fs -chmodg+w /user/hive/warehouse
Command: hive

```
Logging initialized using configuration in jar:file:/home/hadoop/hive-
0.9.0/lib/hive-common-0.9.0.jar!/hive-log4j.properties Hive history
file=/tmp/hadoop/hive_job_log_hadoop_201312121621_1494929084.txt
…….
```

```
hive> show tables;
OK
Time Taken: 2.798 seconds
```

- **Database and table creation,dropping:**

```
hive> CREATE DATABASE [IF NOT EXISTS]
userdb; hive> SHOWDATABASES;
        default
        userdb
hive> DROP DATABASE IF EXISTS userdb;
hive> CREATE TABLE IF NOT EXISTS employee ( eid int, name String,
        > salary String, destinationString)
        > COMMENT„Employeedetails‟
        > ROW FORMATDELIMITED
        > FIELDS TERMINATED BY„\t‟
```

```
> LINES TERMINATED BY,,\n“
> STORED ASTEXTFILE;
```

Example

We will insert the following data into the table. It is a text file named **sample.txt** in **/home/user** directory.

```
1201 Gopal 45000 Technical manager
1202 Manisha 45000 Proof reader
1203 Masthanvali 40000 Technical writer
1204 Krian 40000 Hr Admin
1205 Kranthi 30000 Op Admin
```

```
hive> LOAD DATA LOCAL INPATH '/home/user/sample.txt'
> OVERWRITE INTO TABLEemployee;
hive> SELECT * FROM employee WHERE Salary>=40000;
```

```
+------+--------------+------------+------------------+--------+
|ID    |Name          | Salary|Designation          | Dept|
+------+--------------+------------+------------------+--------+
|1201  | Gopal        | 45000 | Technical manager  | TP     |
|1202  | Manisha      | 45000 | Proofreader        | PR     |
|1203  | Masthanvali  | 40000 | Technical writer   | TP     |
|1204  | Krian        | 40000 | Hr Admin           | HR     |
+------+--------------+------------+------------------+--------+
```

```
hive> ALTER TABLE employee RENAME TO emp;
hive> DROP TABLE IF EXISTS employee;
```

Functions:

Return Type	**Signature**	**Description**
BIGINT	round(double a)	It returns the rounded BIGINT value of the double.
BIGINT	floor(double a)	It returns the maximum BIGINT value that is equal or less than the double.
BIGINT	ceil(double a)	It returns the minimum BIGINT value that is equal or greater than the double.
double	rand(), rand(int seed)	It returns a random number that changes from row to row.
string	concat(string A, string B,...)	It returns the string resulting from concatenating B after A.

string	substr(string A, int start)	It returns the substring of A starting from start position till the end of string A.

string	substr(string A, int start, int length)	It returns the substring of A starting from start position with the given length.
string	upper(string A)	It returns the string resulting from converting all characters of A to upper case.
string	ucase(string A)	Same as above.
string	lower(string A)	It returns the string resulting from converting all characters of B to lower case.

hive> SELECT round(2.6) from temp;
2.0

- **Views:**

Example

Let us take an example for view. Assume employee table as given below, with the fields Id, Name, Salary, Designation, and Dept. Generate a query to retrieve the employee details who earn a salary of more than Rs 30000. We store the result in a view named**emp_30000**.

```
+------+--------------+------------+------------------+--------+
|ID    |Name          | Salary|Designation          |Dept    |
+------+--------------+------------+------------------+--------+
|1201  |Gopal         | 45000 | Technical manager |TP      |
|1202  |Manisha       | 45000 |Proofreader        |PR      |
|1203  | Masthanvali  | 40000 |Technicalwriter    |TP      |
|1204  |Krian         | 40000 |HrAdmin            |HR      |
|1205  |Kranthi       | 30000 |OpAdmin            | Admin|
```

The following query retrieves the employee details using the above scenario:

```
hive> CREATE VIEW emp_30000 AS
> SELECT * FROMemployee
> WHEREsalary>30000;
```

- **Indexes:**

The following query creates an index:

```
hive> CREATE INDEX inedx_salary ON TABLE employee(salary)
> AS'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler';
```

Week – 7: HBase

7) Installation of HBase, Installing thrift along with Practice examples.

Apache HBase Installation Modes

Apache HBase can be installed in three modes.

1) Standalone mode installation (No dependency on Hadoop system)

- This is default mode of HBase
- It runs against local file system
- It doesn't use Hadoop HDFS
- Only HMaster daemon can run
- Not recommended for production environment
- Runs in single JVM

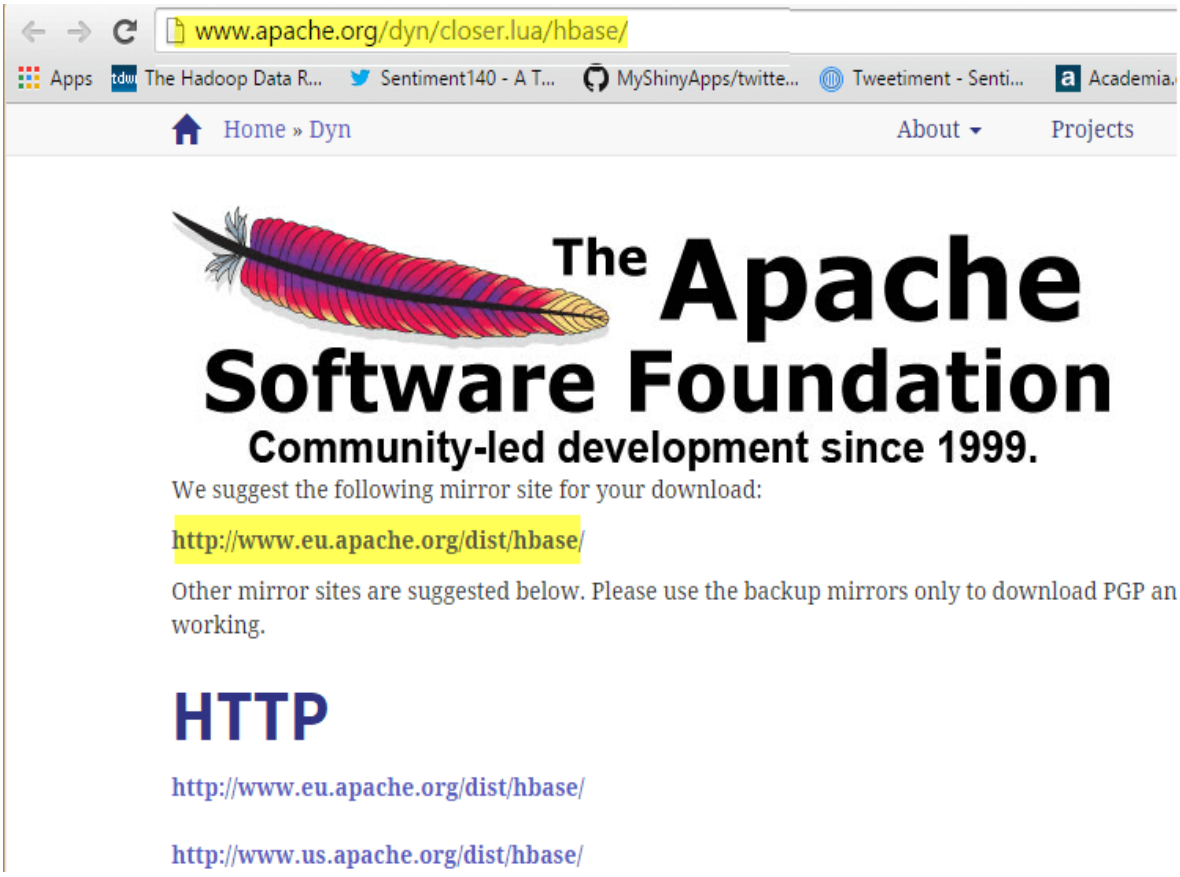
2) Pseudo-Distributed mode installation (Single node Hadoop system + HBase installation)

- It runs on Hadoop HDFS
- All Daemons run in single node
- Recommend for production environment

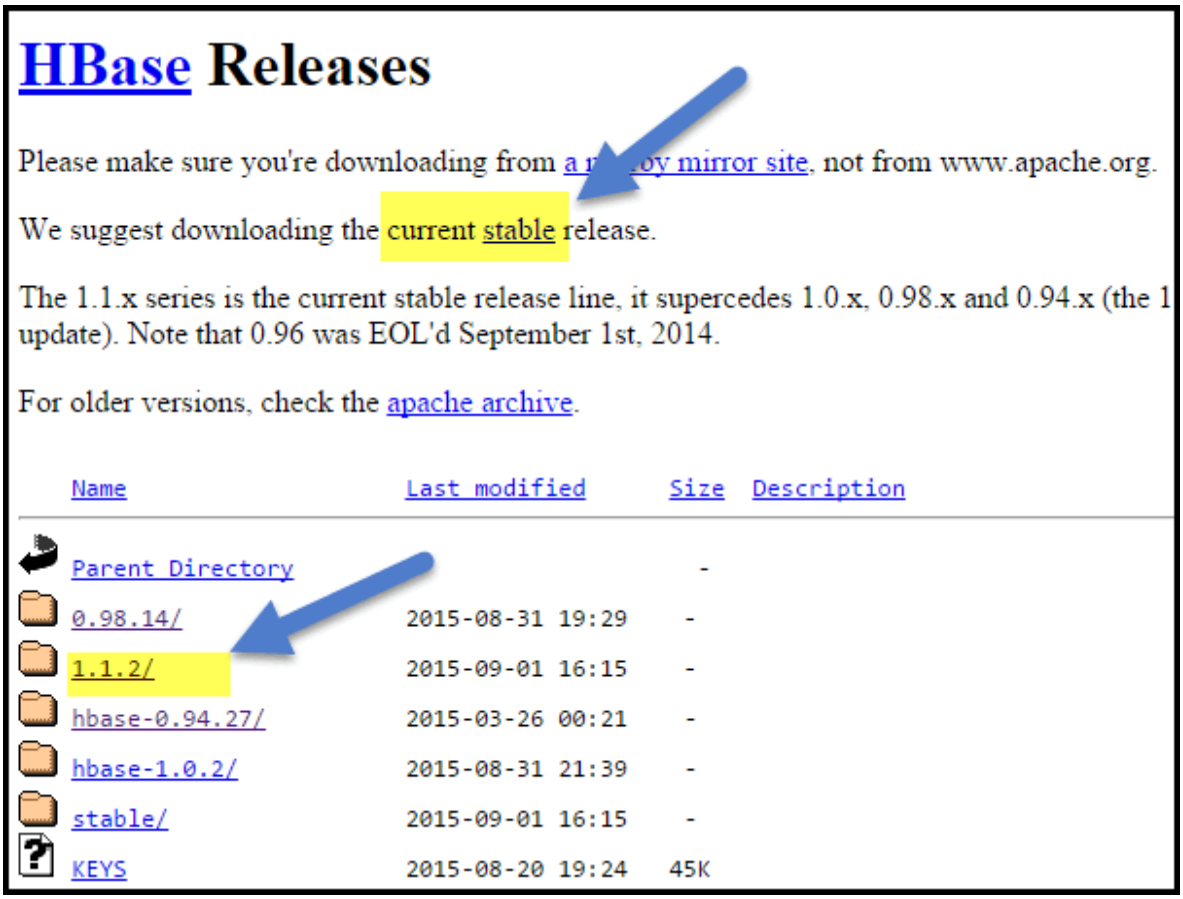
3) Fully Distributed mode installation (MultinodeHadoop environment + HBase installation)

- It runs on Hadoop HDFS
- All daemons going to run across all nodes present in the cluster
- Highly recommended for production environment

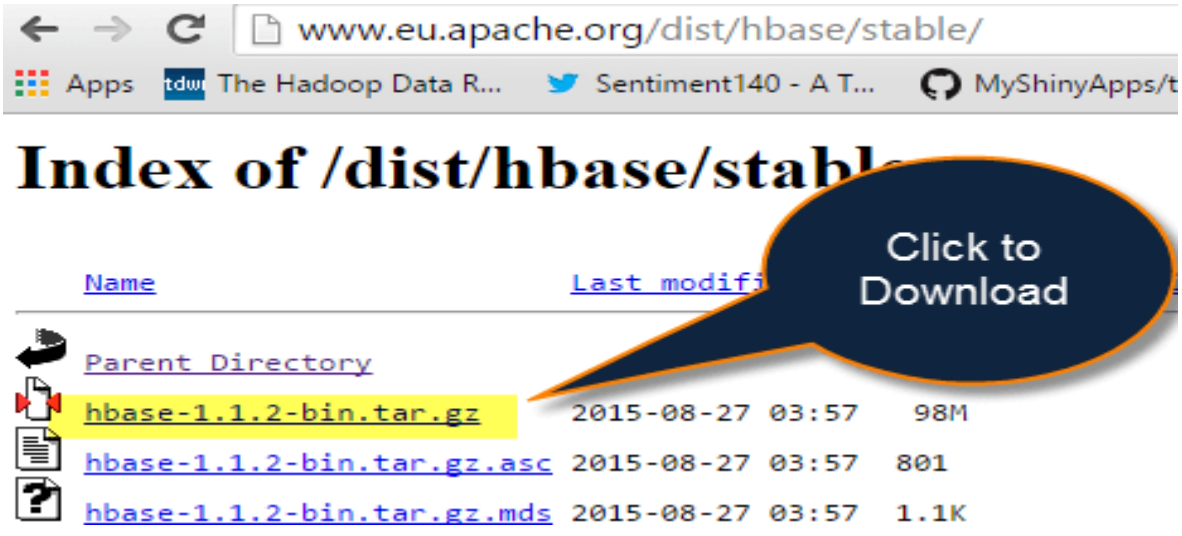
Step 1) Go to the link here to download HBase. It will open a webpage as shown below.



Step 2) Select stable version as shown below 1.1.2 version



Step 3) Click on the hbase-1.1.2-bin.tar.gz. It will download tar file. Copy the tar file into an installation location.



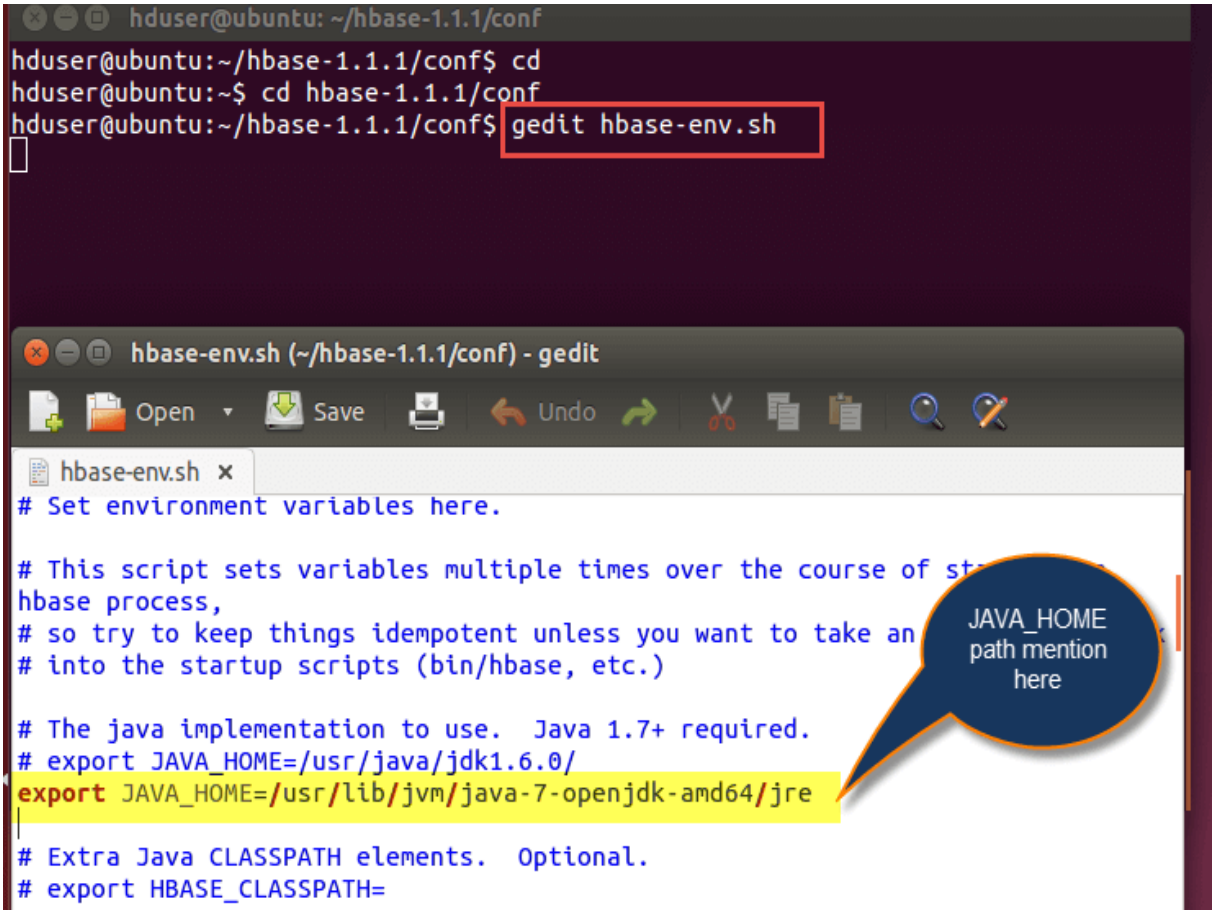
Hbase - Standalone mode installation:

Installation is performed on Ubuntu with Hadoop already installed.

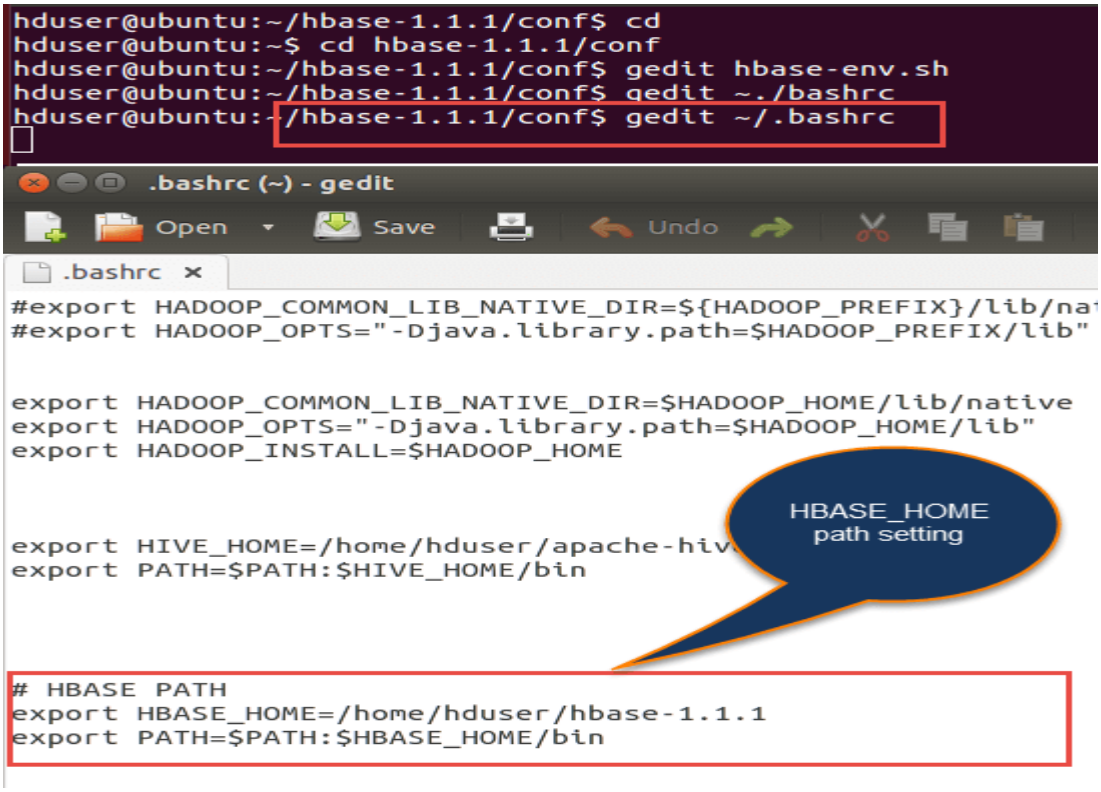
Step 1) Place hbase-1.1.2-bin.tar.gz in /home/hduser

Step 2) Unzip it by executing command $tar -xvf hbase-1.1.2-bin.tar.gz. It will unzip the contents, and it will create hbase-1.1.2 in the location /home/hduser

Step 3) Open hbase-env.sh as below and mention JAVA_HOME path in the location.

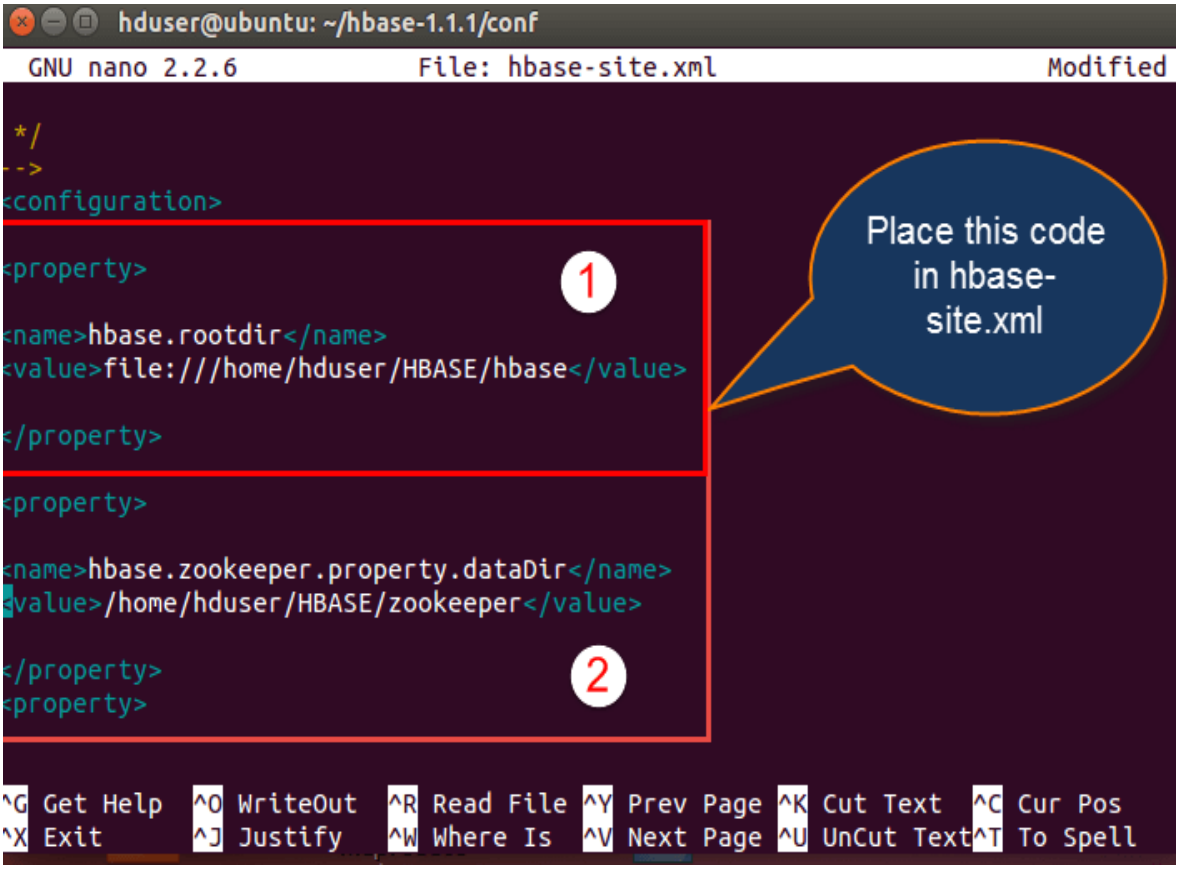


Step 4) Open ~/.bashrc file and mention HBASE_HOME path as shown in below



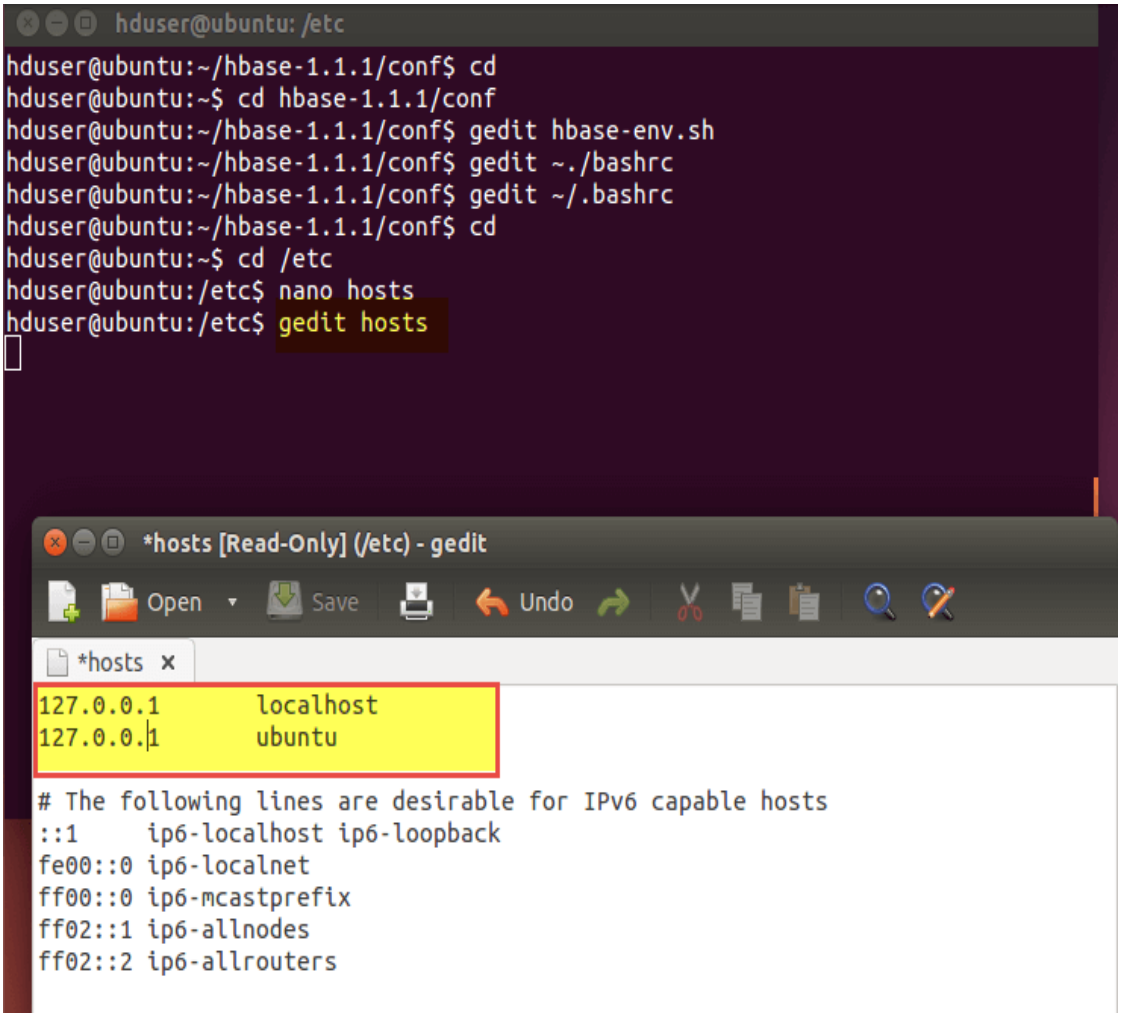
Step 5) Open hbase-site.xml and place the following properties inside the file
hduser@ubuntu$ gedithbase-site.xml(code as below)

```
<property>

<name>hbase.rootdir</name>

<value>file:///home/hduser/HBASE/hbase</value>

</property>

<property>

<name>hbase.zookeeper.property.dataDir</name>

<value>/home/hduser/HBASE/zookeeper</value>

</property>
```



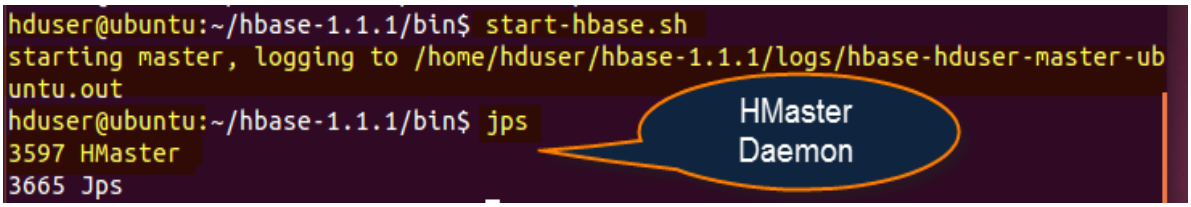
Here we are placing two properties
One for HBase root directory and
Second one for data directory correspond to ZooKeeper.
All HMaster and ZooKeeper activities point out to this hbase-site.xml.

Step 6) Open hosts file present in /etc. location and mention the IPs as shown in below.

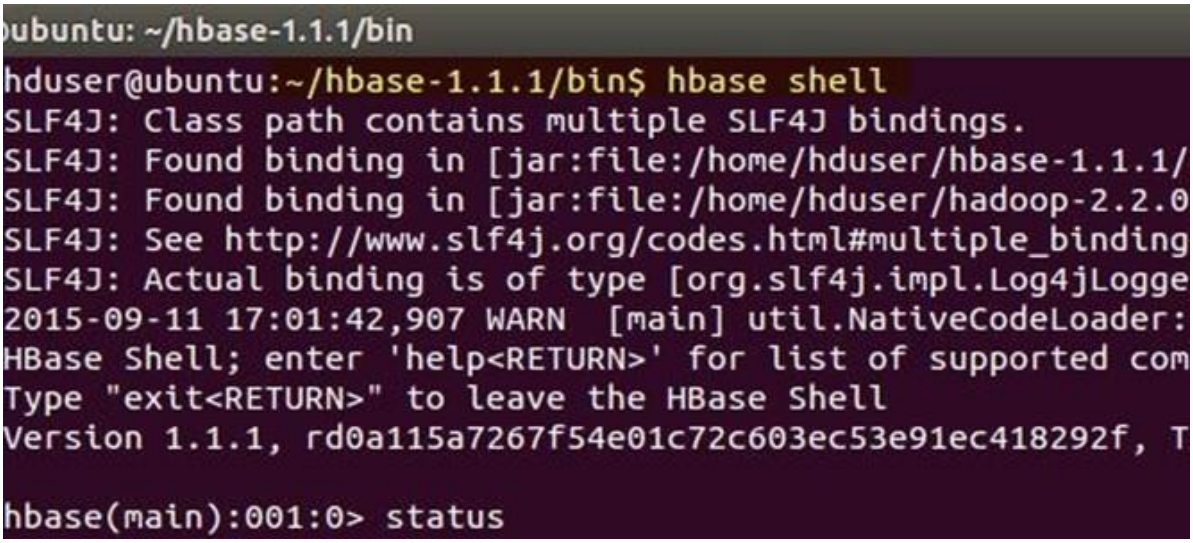


Step 7) Now Run Start-hbase.sh in hbase-1.1.1/bin location as shown below.

And we can check by jps command to see HMaster is running or not.



Step8) HBase shell can start by using "hbase shell" and it will enter into interactive shell mode as shown in below screenshot. Once it enters into shell mode, we can perform all type of commands.



HBase Shell

HBase contains a shell using which you can communicate with HBase. HBase uses the Hadoop File System to store its data. It will have a master server and region servers. The data storage will be in the form of regions (tables). These regions will be split up and stored in region servers.

The master server manages these region servers and all these tasks take place on HDFS. Given below are some of the commands supported by HBase Shell.

General Commands

status - Provides the status of HBase, for example, the number of servers.

version - Provides the version of HBase being used.

table_help - Provides help for table-reference commands.

whoami - Provides information about the user.

Data Definition Language

These are the commands that operate on the tables in HBase.

create - Creates a table.

list - Lists all the tables in HBase.

disable - Disables a table.

is_disabled - Verifies whether a table is disabled.

enable - Enables a table.

is_enabled - Verifies whether a table is enabled.

describe - Provides the description of a table.

alter - Alters a table.

exists - Verifies whether a table exists.

drop - Drops a table from HBase**.**

drop_all - Drops the tables matching the 'regex' given in the command.

Java Admin API - Prior to all the above commands, Java provides an Admin API to achieve DDL functionalities through programming. Under org.apache.hadoop.hbase.client package, HBaseAdmin and HTableDescriptor are the two important classes in this package that provide DDL functionalities.

Data Manipulation Language

put - Puts a cell value at a specified column in a specified row in a particular table.

get - Fetches the contents of row or a cell.

delete - Deletes a cell value in a table.

deleteall - Deletes all the cells in a given row.

scan - Scans and returns the table data.

count - Counts and returns the number of rows in a table.

truncate - Disables, drops, and recreates a specified table.

Week – 8: R Programming

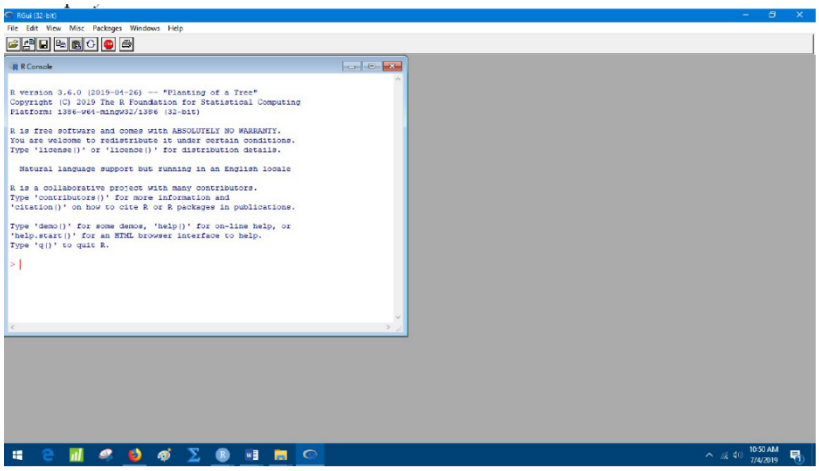
8) Installation of R, along with Practice examples in R.

Installation of R software in Windows and Linux environments

Requirements Analysis

Installation of R in Windows OS: The Comprehensive R Archive Network (CRAN) is a network ofwebsites that host the R program and that mirror the original R website. The benefit of having this networkof websites is improved download speeds. For all intents and purposes, CRAN is the R website and holdsdownloads (including old versions of software) and documentation. R can be installed inWindows7/8/10/Vista and supports both the 32-bit and 64-bit versions. Go to the CRAN website and selectthe latest installer R 3.4.0 for Windows and download the .exe file. Double click on the download file andselect Run as Administrator form the popup menu. Select the language to be used for installation and followthe directions. The installation folder for R can be found in C:\Programs\R. The steps for installing R:

1. Click on the link https://cran.r-project.org/bin/windows/base/ which redirects you to the download. page.
2. Select the latest installer R-3.4.0 for installation and download the same. After download, clicking. on the setup file opens the dialog box.
3. Click on the 'Next' button starts the installation process. This redirects you to the license window. and selecting 'Next'.
4. After selecting the Next button from the previous step the installation folder path is required. Selectthe desired folder for installation; it is advisable to select the C directory for smooth running of theprogram.
5. Next select the components for installation based on the requirements of your operating system toavoid unwanted use of disk space.
6. In the next dialog box, we need to select the start menu folder. Here, it is better to go with thedefault option given by the installer.
7. After setting up the Start menu folder, check the additional options for completing the setup.
8. After clicking next from the previous step, the installation procedureend and the window isdisplayed. Click 'Finish' to exist from the installation window.



Installing R-Studio

Installing and Configuring R-Studio in Windows: The Integrated Development Environment(IDE) for R is R Studio and it provides a variety of features such as an editor with direct code execution and syntax highlighting, a console, tools for plotting graphs, history lookup, debugging, and an environment for workspace creation. R Studio can be installed in any of the Windows platforms such

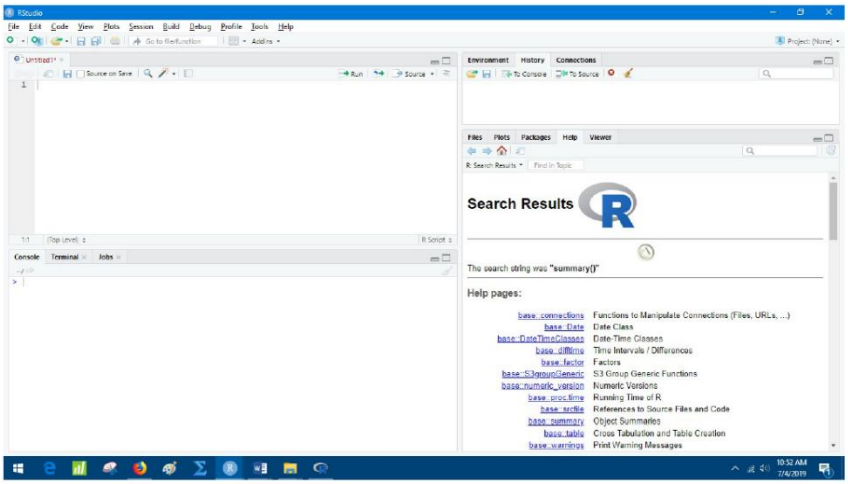
as Windows 7/8/10/Vista and can be configured within a few minutes. The basic requirement is R 2.11.1+ version. The following are the steps involved to setup R Studio:

1) Download the latest version of R Studio just by clicking on the link provided here https://www.rstudio.com/products/rstudio/download/ and it redirects you to download page. There are two versions of R Studio available – desktop and server. Based on your usage and comfort, select the appropriate version to initiate your download. The latest desktop version for R Studio is 1.0.136.

2) Download the .exe file and double click on it to initiate the installation.

3) Click on the 'Next' button and it redirects you to select the installation folder. Select 'C:\' as your installation directory since R and R Studio must be installed in the same directory to avoid path issues for running R programs.

4) Click 'Next' to continue and a dialog box asking you to select the Start menu folder opens. It is advisable to create your own folder to avoid any possible confusion and click on Install button to install R Studio. After completion of installation, clicking 'Next' from the previous step, the installation procedure ends, and the window is displayed. Click 'Finish' to exist from the installation window.



Installation of R in Ubuntu

Installation of R in Ubuntu: Go to software center and search for R Base andinstall. Then open terminaland enter R to get R command prompt in terminal.Installation of R-studio in Ubuntu: Open terminal and type the followingcommands.

```
sudo add-apt-repository 'deb https://ftp.ussg.iu.edu/CRAN/bin/linux/ubuntu trusty/'
sudo apt-get update
sudo apt-get install r-base
sudo apt-get install r-base-dev

# Download and Install RStudio
sudo apt-get install gdebi-core
wget https://download1.rstudio.org/rstudio-1.0.44-amd64.deb
sudo gdebi rstudio-1.0.44-amd64.deb
rm rstudio-1.0.44-amd64.deb
```

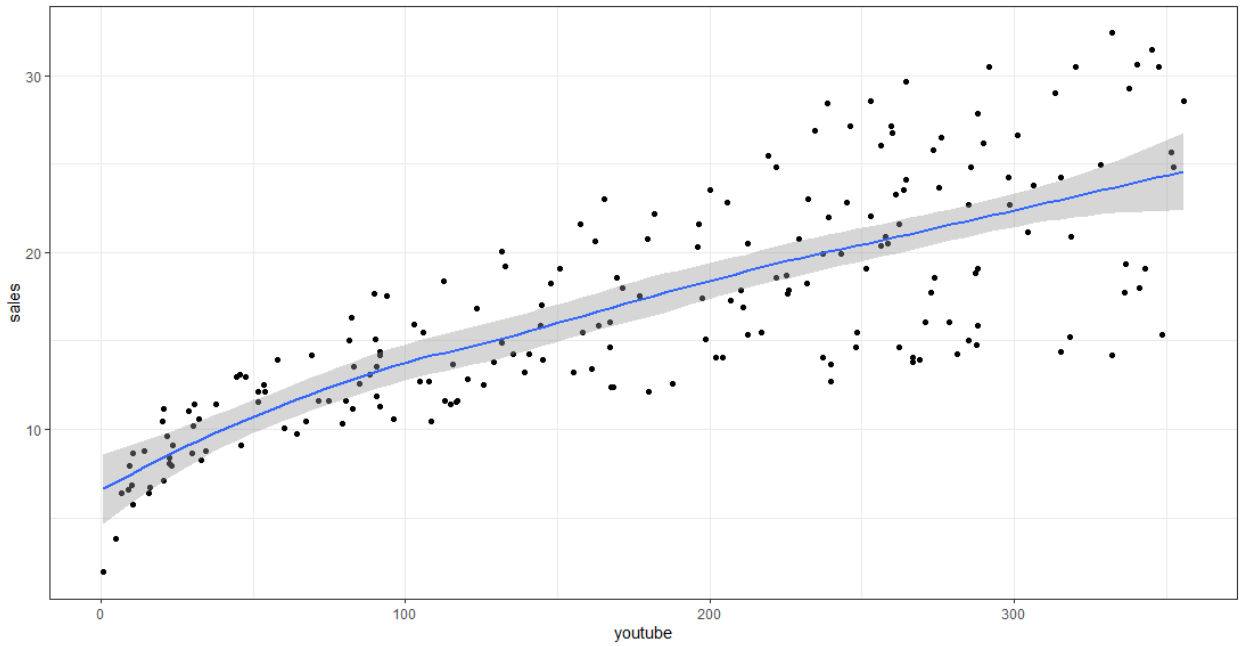
Linear Regression :

```
library(tidyverse)
library(caret)
theme_set(theme_bw())
data("marketing", package = "datarium")
```

```
sample_n(marketing, 3)
set.seed(123)
training.samples<- marketing$sales %>%
createDataPartition(p = 0.8, list = FALSE)
train.data<- marketing[training.samples, ]
test.data<- marketing[-training.samples, ]
model <- lm(sales ~., data = train.data)
summary(model)
model <- lm(sales ~ youtube, data = train.data)
summary(model)$coef
model <- lm(sales ~ facebook, data = train.data)
summary(model)$coef
newdata<- data.frame(youtube = c(0,  1000))
model %>% predict(newdata)
model <- lm(sales ~ youtube + facebook + newspaper, data = train.data)
summary(model)$coef
ggplot(marketing, aes(x = youtube, y = sales)) +geom_point() +stat_smooth()
predictions <- model %>% predict(test.data)
RMSE(predictions, test.data$sales)
R2(predictions, test.data$sales)
```

Results

```
Call:
lm(formula = sales ~ ., data = train.data)
Residuals:
    Min      1Q  Median      3Q     Max
-10.7142  -0.9939   0.3684   1.4494   3.3619
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.594142   0.420815   8.541 1.05e-14 ***
youtube     0.044636   0.001552  28.758< 2e-16 ***
facebook    0.188823   0.009529  19.816< 2e-16 ***
newspaper   0.002840   0.006442   0.441     0.66
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 2.043 on 158 degrees of freedom
Multiple R-squared:  0.8955,  Adjusted R-squared:  0.8935
F-statistic: 451.2 on 3 and 158 DF,  p-value: < 2.2e-16
```


sales
youtube
30
20
10
0
100
200
300

Week – 9

(Additional Program on Mapreduce)

9) Write a MapReduce program that mines weather data.

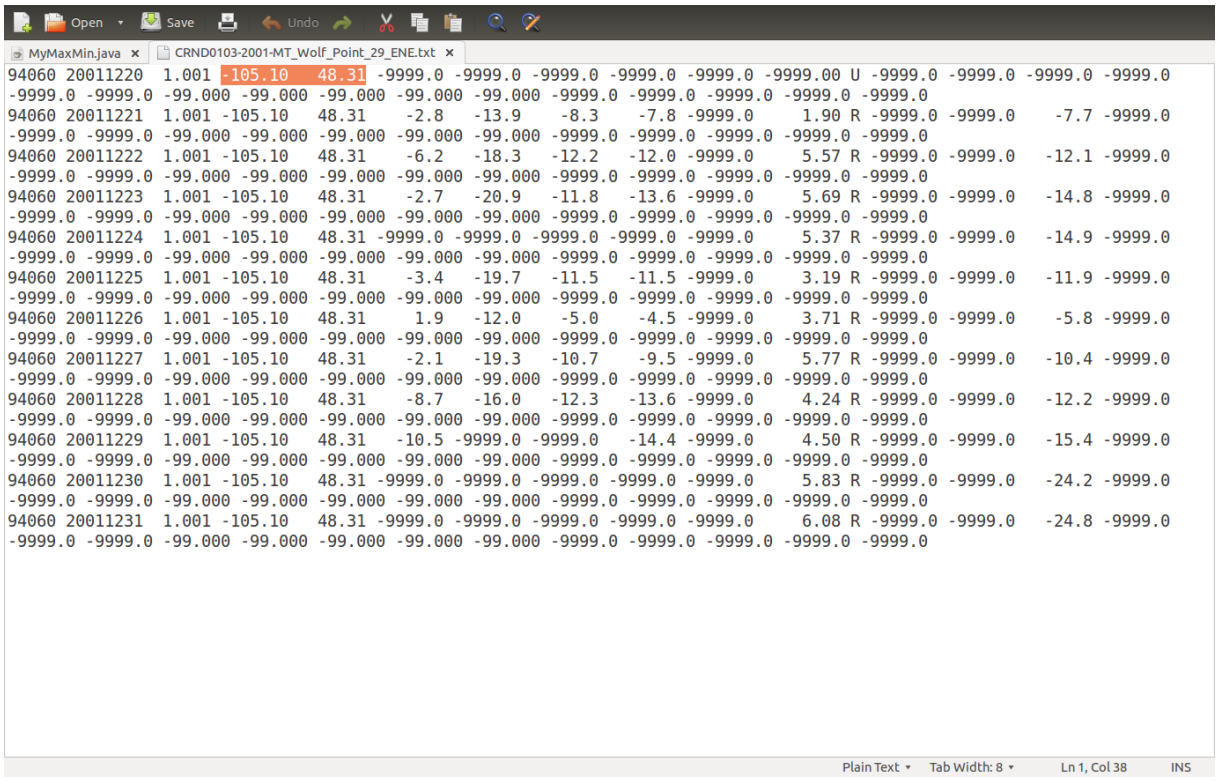
Weather sensors collecting data every hour at many locations across the globe gather a large volume of log data, which is good candidate for analysis with MapReduce, since it is semi-structured and record-oriented.

PROGRAM:

```
import java.io.IOException;
import java.util.Iterator;
importorg.apache.hadoop.fs.Path;
importorg.apache.hadoop.io.LongWritable;
importorg.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
public class MyMaxMin {
 public static class MaxTemperatureMapper extends Mapper<LongWritable, Text, Text,
Text> {
@Override
public void map(LongWritable arg0, Text Value, Context context) throws IOException,
InterruptedException {
                String line = Value.toString();
                if (!(line.length() == 0)) {
                  String date = line.substring(6, 14);
                   float temp_Min = Float.parseFloat(line.substring(22, 28).trim());
                  float temp_Max = Float.parseFloat(line.substring(32, 36).trim());
                  if (temp_Max> 35.0) {
                context.write(new Text("Hot Day " + date),new
Text(String.valueOf(temp_Max)));
                  }
                  if (temp_Min< 10) {
                        context.write(new Text("Cold Day " + date),new
Text(String.valueOf(temp_Min)));
                  }
                }
        }
 }
 public static class MaxTemperatureReducer extends Reducer<Text, Text, Text, Text>
 {
  public void reduce(Text Key, Iterator<Text> Values, Context context)throws
IOException, InterruptedException
```

```
 {
                String temperature = Values.next().toString();
                context.write(Key, new Text(temperature));
        }
}
public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = new Job(conf, "weather example");
        job.setJarByClass(MyMaxMin.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);
        job.setMapperClass(MaxTemperatureMapper.class);
        job.setReducerClass(MaxTemperatureReducer.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        Path OutputPath = new Path(args[1]);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 :1);
 }
}
```

sample input dataset:

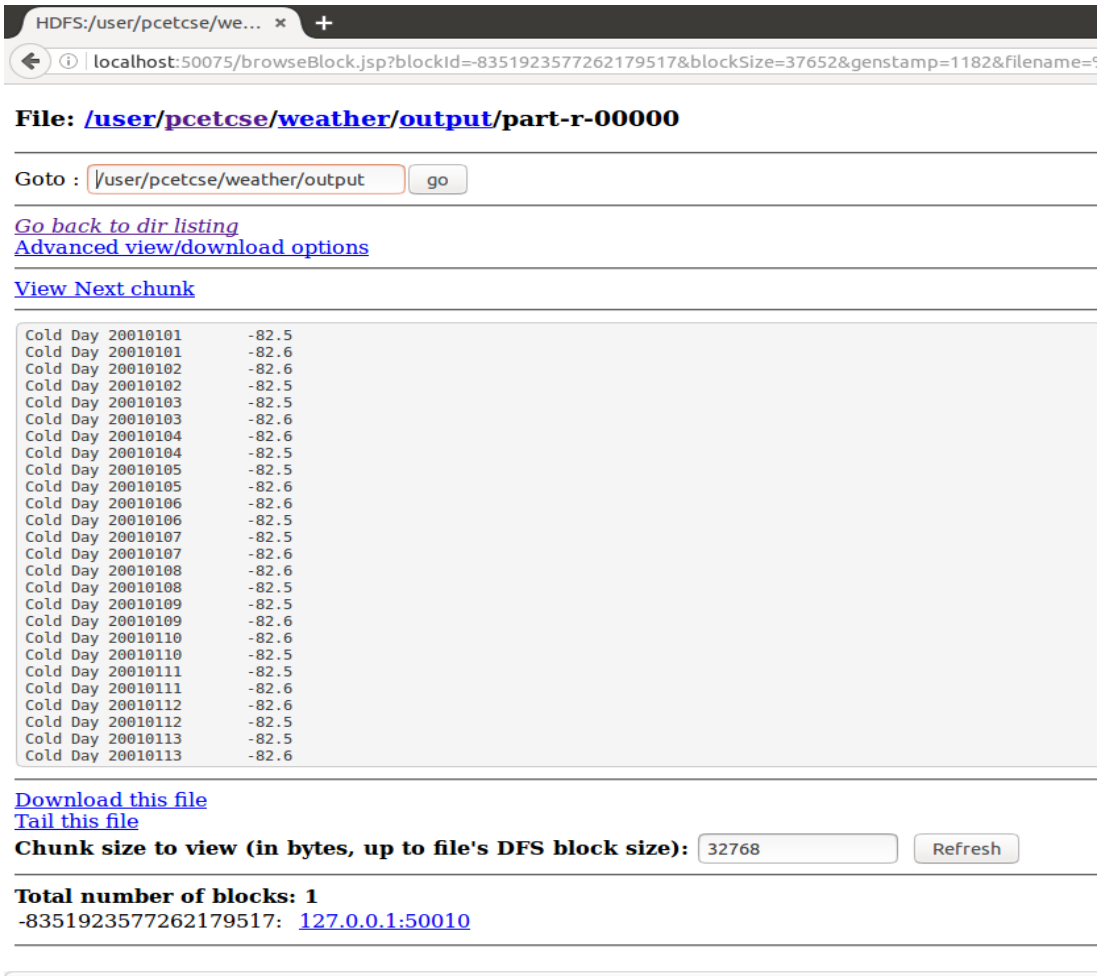


- Compiling and creating jar file for hadoopmapreducejavaprogram:
  Command: hadoopcom.sun.tools.javac.Main MyMaxMin.java
  Command: jar cvf we.jarMyMaxMin*.class

- Runnning weather dataset mapreduce jar file onhadoop
  Command: hadoop jar we.jar MyMaxMin weather/input weather/output

```
pcetcse@pcetcse1:~/weatherdata$ hadoop com.sun.tools.javac.Main MyMaxMin.java
pcetcse@pcetcse1:~/weatherdata$ jar cvf we.jar MyMaxMin*.class
added manifest
adding: MyMaxMin.class(in = 1642) (out= 840)(deflated 48%)
adding: MyMaxMin$MaxTemperatureMapper.class(in = 2029) (out= 882)(deflated 56%)
adding: MyMaxMin$MaxTemperatureReducer.class(in = 1283) (out= 533)(deflated 58%)
pcetcse@pcetcse1:~/weatherdata$ hadoop jar we.jar MyMaxMin weather/input/ weather/output
08/01/01 00:29:19 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments.
08/01/01 00:29:19 INFO input.FileInputFormat: Total input paths to process : 8
08/01/01 00:29:19 INFO util.NativeCodeLoader: Loaded the native-hadoop library
08/01/01 00:29:19 WARN snappy.LoadSnappy: Snappy native library not loaded
08/01/01 00:29:20 INFO mapred.JobClient: Running job: job_200801010014_0003
08/01/01 00:29:21 INFO mapred.JobClient:  map 0% reduce 0%
08/01/01 00:29:53 INFO mapred.JobClient:  map 25% reduce 0%
08/01/01 00:30:26 INFO mapred.JobClient:  map 50% reduce 0%
08/01/01 00:30:33 INFO mapred.JobClient:  map 50% reduce 16%
```

output:



Week – 10

(Additional Program on Mapreduce)

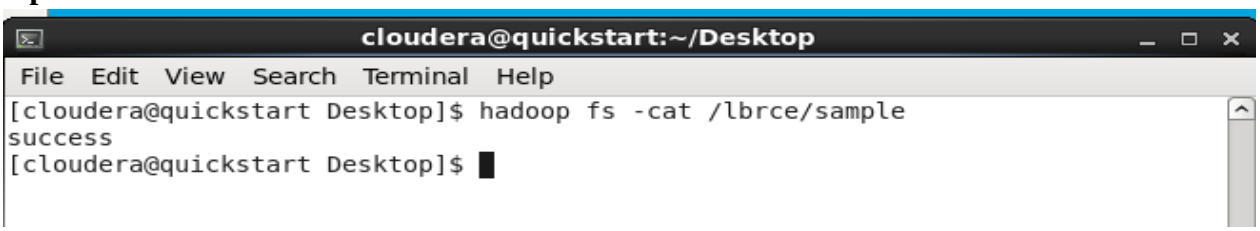
10) Write a program to group of repeated characters in a string by using Mapreduce.

PROGRAM:

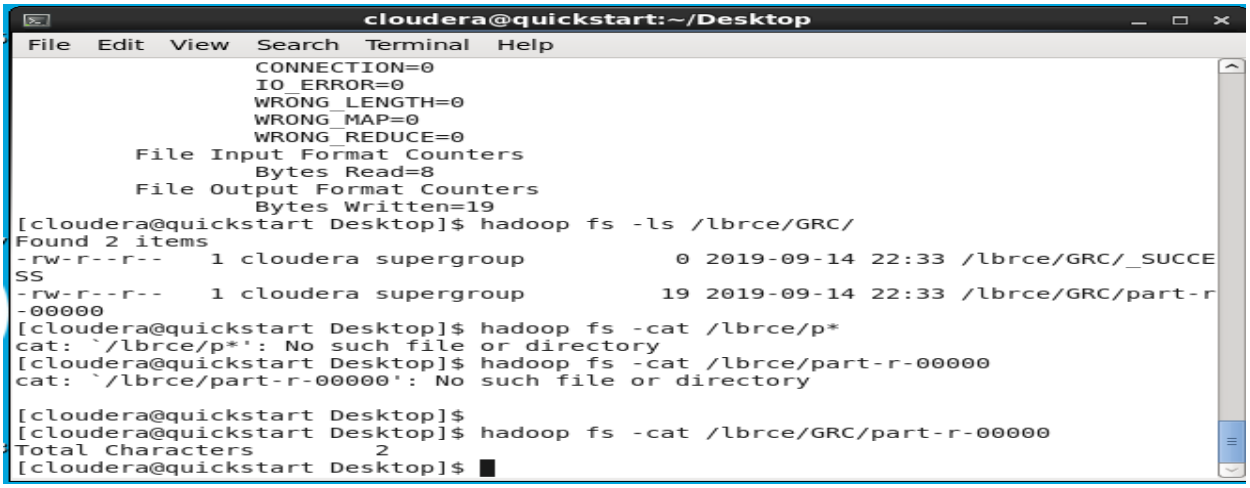
```
package com.lbrce.charcount;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class GroupCharCountMapperextends Mapper<LongWritable, Text, Text, IntWritable>{
  @Override
  public void map(LongWritable key, Text value, Context con)
     throws IOException, InterruptedException{
    String line = value.toString();
    char[] chars = line.toCharArray();
    for(int i=0;i<chars.length-1;i++) {
if(chars[i]==chars[i+1])
          con.write(new Text("Total Characters"), new IntWritable(1));
    }
  }
}
package com.lbrce.charcount;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class GroupCharCountReducerextends Reducer<Text, IntWritable, Text, IntWritable> {
  /*
   * bigdata<1>
   */
  @Override
  public void reduce(Text key, Iterable<IntWritable>values,Context con)
     throws IOException, InterruptedException {
          int sum = 0;
          for(IntWritable i:values) {
                sum = sum + i.get();
          }
          con.write(key, new IntWritable(sum));
     }
}
/*
 * grouping
 * map(key,List<IntWritable>)
 *
```

```
 */
package com.lbrce.charcount;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class GroupCharCountDriver {
  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: Group Char Count <input path><output path>");
      System.exit(-1);
    }
    Job job = new Job();
    job.setJobName("Group Char Count");
    job.setJarByClass(GroupCharCountDriver.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.setMapperClass(GroupCharCountMapper.class);
    job.setReducerClass(GroupCharCountReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Input:



Output:



Week – 11

(Additional Program on Mapreduce)

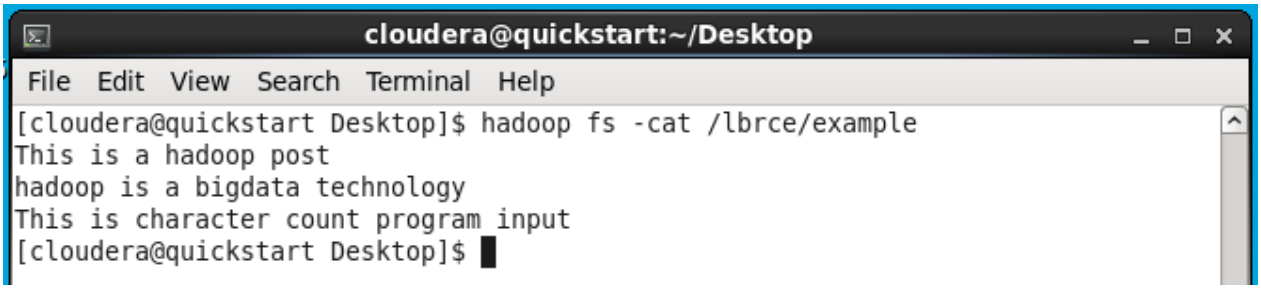
11) Write a program to count characters in a text data by using Mapreduce.

PROGRAM:

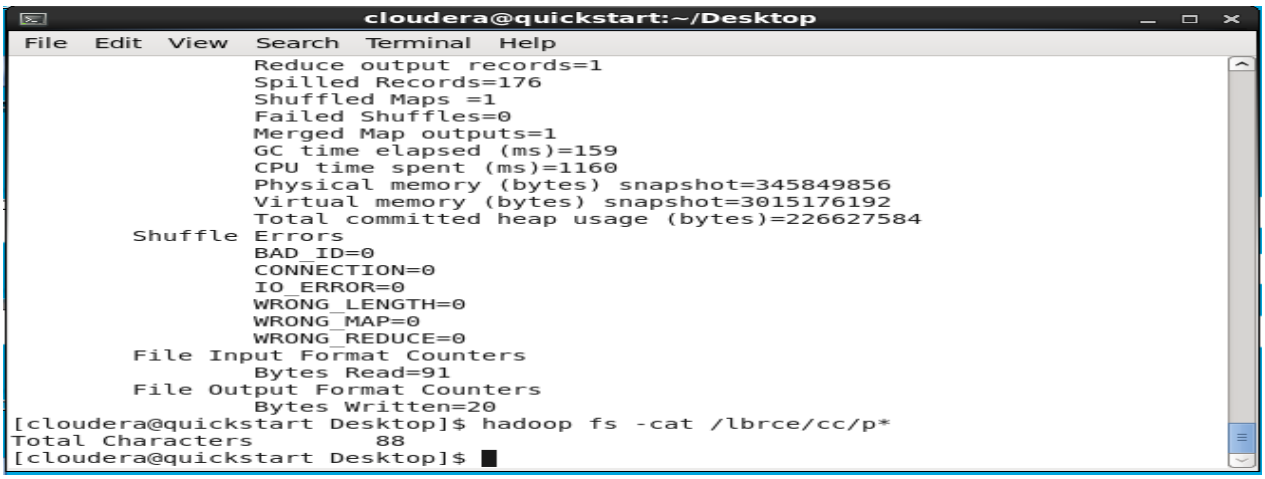
```
package com.lbrce.charcount;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class CharCountMapperextends Mapper<LongWritable, Text, Text, IntWritable>{
 @Override
 public void map(LongWritable key, Text value, Context con)
    throws IOException, InterruptedException{
  String line = value.toString();
  char[] chars = line.toCharArray();
  for(char c:chars) {
        con.write(new Text("Total Characters"), new IntWritable(1));
  }
 }
}
package com.lbrce.charcount;
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class CharCountReducerextends Reducer<Text, IntWritable, Text, IntWritable> {
 /*
  * bigdata<1>
  */
 @Override
 public void reduce(Text key, Iterable<IntWritable>values,Context con)
    throws IOException, InterruptedException {
        int sum = 0;
        for(IntWritable i:values) {
                sum = sum + i.get();
        }
        con.write(key, new IntWritable(sum));
    }
}
/*
 * grouping
 * map(key,List<IntWritable>)
 *
 */
```

```
package com.lbrce.charcount;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class CharCountDriver {
  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: CharCount<input path><output path>");
      System.exit(-1);
    }
    Job job = new Job();
    job.setJobName("CharCount");
    job.setJarByClass(CharCountDriver.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    job.setMapperClass(CharCountMapper.class);
    job.setReducerClass(CharCountReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Input:



Output:



Week – 12

(Additional Program on Mapreduce)

12) Write a program to find how many flights between origin and destination by using Mapreduce.

PROGRAM:

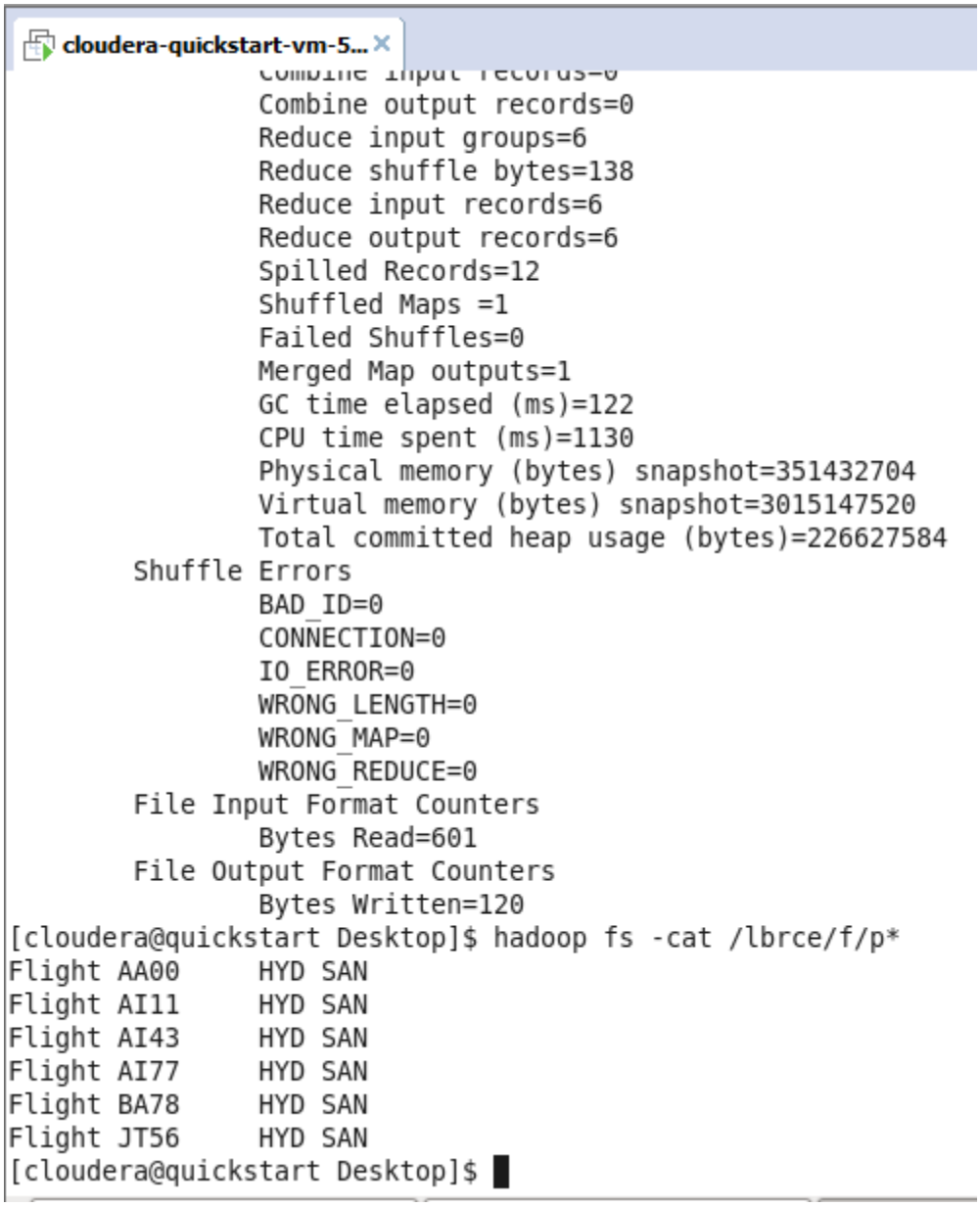
```
package com.lbrce.flight;
import java.io.IOException;
import java.util.Iterator;
importorg.apache.hadoop.fs.Path;
importorg.apache.hadoop.io.LongWritable;
importorg.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
public class Flight {
public static class FlightMapper extends Mapper<LongWritable, Text, Text, Text>
{
public void map(LongWritable arg0, Text Value, Context context) throws IOException,
InterruptedException
{
String line = Value.toString();
if (!(line.length() == 0))
{
String fno = line.substring(0, 4);
String origin=line.substring(8, 12).trim();
String dest =line.substring(13, 18).trim();
if(origin.equals("HYD")&&dest.equals("SAN"))
{
context.write(new Text("Flight " + fno),new Text("HYD SAN"));
}
}
}
}
public static class FlightReducer extends Reducer<Text, Text, Text, Text>
{
public void reduce(Text Key, Iterator<Text> Values, Context context)throws IOException,
InterruptedException
{
String nof = Values.next().toString();
context.write(Key, new Text(nof));
} }
```

```
public static void main(String[] args) throws Exception
{
Configuration conf = new Configuration();
Job job = new Job(conf, "weather example");
job.setJarByClass(Flight.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);
job.setMapperClass(FlightMapper.class);
job.setReducerClass(FlightReducer.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
Path OutputPath = new Path(args[1]);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ? 0 :1);
}
}
```

Input:

FlightNum	**Origin**	**Destination**	**ArrivalTime**
AI111	**HYD**	**SAN**	**22:30**
QA222	**BOM**	**NEY**	**24:26**
SA333	**DEL**	**DAL**	**32:24**
BA444	**CHE**	**SAN**	**42:15**
SA555	**HYD**	**NEJ**	**24:26**
QA666	**BAN**	**DAL**	**22:30**
AI777	**HYD**	**SAN**	**32:24**
SA888	**DEL**	**SAN**	**42:15**
BA999	**BAN**	**NEY**	**32:24**
SA123	**BOM**	**NEJ**	**24:26**
QA321	**CHE**	**SAN**	**42:15**
SA345	**BAN**	**DAL**	**24:26**
AI456	**CHE**	**SAN**	**42:15**
BA789	**HYD**	**SAN**	**22:30**
QA156	**BOM**	**NEJ**	**32:24**
SA234	**BAN**	**DAL**	**24:26**
BA132	**BOM**	**NEJ**	**42:15**
AI431	**HYD**	**SAN**	**22:30**
AA001	**CHE**	**SAN**	**32:24**
AA007	**BOM**	**NEJ**	**24:26**
AA009	**HYD**	**SAN**	**24:26**
DT876	**BAN**	**DAL**	**42:15**
JT567	**HYD**	**SAN**	**22:30**

Output:

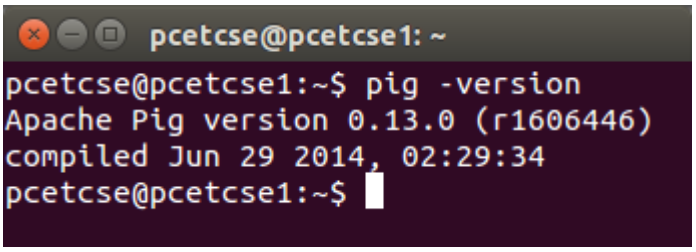

cloudera-quickstart-vm-5...
Combine input records=0
Combine output records=0
Reduce input groups=6
Reduce shuffle bytes=138
Reduce input records=6
Reduce output records=6
Spilled Records=12
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=122
CPU time spent (ms)=1130
Physical memory (bytes) snapshot=351432704
Virtual memory (bytes) snapshot=3015147520
Total committed heap usage (bytes)=226627584
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=601
File Output Format Counters
Bytes Written=120
[cloudera@quickstart Desktop]$ hadoop fs -cat /lbrce/f/p*
Flight AA00 HYD SAN
Flight AI11 HYD SAN
Flight AI43 HYD SAN
Flight AI77 HYD SAN
Flight BA78 HYD SAN
Flight JT56 HYD SAN
[cloudera@quickstart Desktop]$

Week – 13: Pig

Install and Run Pig then write Pig Latin scripts to sort, group, join, project, and filter your data.

PROCEDURE:

- Download and extractpig-0.13.0.
  Command: wgethttps://archive.apache.org/dist/pig/pig-0.13.0/pig-0.13.0.tar.gz
  Command: tar xvf pig-0.13.0.tar.gz
  Command: sudo mv pig-0.13.0 /usr/lib/pig
- Set Path forpig
  Command: sudogedit
  $HOME/.bashrc export
  PIG_HOME=/usr/lib/pig
  export PATH=$PATH:$PIG_HOME/bin
  export PIG_CLASSPATH=$HADOOP_COMMON_HOME/conf
- pig.propertiesfile
  In the conf folder of Pig, we have a file named pig.properties. In the pig.properties file, you can set various parameters as given below.
  pig -h properties
- **Verifying theInstallation**
  Verify the installation of Apache Pig by typing the version command. If the installation is successful, you will get the version of Apache Pig as shown below.
  Command: pig -version



Local mode	MapReduce mode
Command: $ pig -x local 15/09/28 10:13:03 INFO pig.Main: Logging error messages to: /home/Hadoop/pig_1443415383991.l og 2015-09-28 10:13:04,838 [main] INFO org.apache.pig.backend.hadoop.execution engine.HExecutionEngine - Connecting to hadoop file system at: file:/// grunt>	Command: $ pig -x mapreduce 15/09/28 10:28:46 INFO pig.Main: Logging error messages to: /home/Hadoop/pig_1443416326123.l og 2015-09-28 10:28:46,427 [main] INFO org.apache.pig.backend.hadoop.executi on engine.HExecutionEngine - Connecting to hadoopfile system at: file:/// grunt>

Grouping Of Data:

- put dataset intohadoop

  Command: hadoop fs -put pig/input/data.txt pig_data/

- Run pig script program of GROUP on hadoopmapreduce

grunt>

```
student_details = LOAD
'hdfs://localhost:8020/user/pcetcse/pig_data/student_details.txt' USING PigStorage(',')
as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray,
city:chararray);
group_data = GROUP student_details by age;
Dump group_data;
```

Output:

Joining Of Data:

- Run pig script program of JOIN on hadoopmapreduce

grunt>

```
customers = LOAD 'hdfs://localhost:8020/user/pcetcse/pig_data/customers.txt'
USING PigStorage(',')as (id:int, name:chararray, age:int, address:chararray,
salary:int);
orders = LOAD 'hdfs://localhost:8020/user/pcetcse/pig_data/orders.txt' USING
PigStorage(',')as (oid:int, date:chararray, customer_id:int, amount:int);
```

grunt>coustomer_orders = JOIN customers BY id, orders BY customer_id;

- **Verification**

  Verify the relation **coustomer_orders**using the **DUMP** operator as shown below.

  grunt>Dump coustomer_orders;

- **Output**

  You will get the following output that wills the contents of the relation named **coustomer_orders**.

  (2,Khilan,25,Delhi,1500,101,2009-11-2000:00:00,2,1560)
  (3,kaushik,23,Kota,2000,100,2009-10-0800:00:00,3,1500)
  (3,kaushik,23,Kota,2000,102,2009-10-0800:00:00,3,3000)
  (4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)

Sorting of Data:

- Run pig script program of SORT on hadoopmapreduce

  Assume that we have a file named **student_details.txt** in the HDFS directory **/pig_data/** as shown below.

  student_details.txt

  001,Rajiv,Reddy,21,9848022337,Hyderabad
  002,siddarth,Battacharya,22,9848022338,Kolkata
  003,Rajesh,Khanna,22,9848022339,Delhi
  004,Preethi,Agarwal,21,9848022330,Pune
  005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
  006,Archana,Mishra,23,9848022335,Chennai
  007,Komal,Nayak,24,9848022334,trivendram
  008,Bharathi,Nambiayar,24,9848022333,Chennai

  And we have loaded this file into Pig with the schema name **student_details**as shown below.

  grunt>

  student_details = LOAD „hdfs://localhost:8020/user/pcetcse/pig_data/student_details.txt' USING PigStorage(',')as (id:int, firstname:chararray, lastname:chararray,age:int, phone:chararray,city:chararray);

  Let us now sort the relation in a descending order based on the age of the student and store it into another relation named **data** using the **ORDER BY** operator as shown below.

  grunt>order_by_data= ORDER **student_details**BY age DESC;

- **Verification**

  Verify the relation **order_by_data**using the **DUMP** operator as shown below.

  grunt>Dump **order_by_**data;

- **Output**

It will produce the following output, displaying the contents of the relation **order_by_data**as follows.

(8,Bharathi,Nambiayar,24,9848022333,Chennai)

(7,Komal,Nayak,24,9848022334,trivendram)
(6,Archana,Mishra,23,9848022335,Chennai)
(5,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar)
(3,Rajesh,Khanna,22,9848022339,Delhi)
(2,siddarth,Battacharya,22,9848022338,Kolkata)
(4,Preethi,Agarwal,21,9848022330,Pune)
(1,Rajiv,Reddy,21,9848022337,Hyderabad)

Filtering of data:

- Run pig script program of FILTER on hadoopmapreduce

Assume that we have a file named **student_details.txt** in the HDFS directory **/pig_data/** as shown below.

student_details.txt

001,Rajiv,Reddy,21,9848022337,Hyderabad
002,siddarth,Battacharya,22,9848022338,Kolkata
003,Rajesh,Khanna,22,9848022339,Delhi
004,Preethi,Agarwal,21,9848022330,Pune
005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar
006,Archana,Mishra,23,9848022335,Chennai
007,Komal,Nayak,24,9848022334,trivendram
008,Bharathi,Nambiayar,24,9848022333,Chennai

And we have loaded this file into Pig with the schema name **student_details**as shown below.

grunt>
student_details = LOAD
„hdfs://localhost:8020/user/pcetcse/pig_data/student_details.txt' USING PigStorage(',')as (id:int, firstname:chararray, lastname:chararray,age:int, phone:chararray,city:chararray);

Let us now use the Filter operator to get the details of the students who belong to the city Chennai.

grunt>filter_data = FILTER student_details BY city == 'Chennai';

- **Verification**

Verify the relation **filter_data**using the **DUMP** operator as shown below.

grunt>Dump filter_data;

- **Output**

It will produce the following output, displaying the contents of the relation **filter_data**as follows.

(6,Archana,Mishra,23,9848022335,Chennai)
(8,Bharathi,Nambiayar,24,9848022333,Chennai)

VIVA – QUESTIONS

1. What are the basic differences between relational database and HDFS?
2. Explain "Big Data" and what are five V's of Big Data?
3. What is Hadoop and its components.
4. What are HDFS and YARN?
5. Tell me about the various Hadoop daemons and their roles in a Hadoop cluster.
6. Compare HDFS with Network Attached Storage (NAS).
7. List the difference between Hadoop 1 and Hadoop2.
8. What are active and passive "Name nodes"?
9. Why does one remove or add nodes in a Hadoop cluster frequently?
10. What happens when two clients try to access the same file in the HDFS?
11. How does Name node tackle Data node failures?
12. **What will you do when Name node is down?**
13. What is a check point?
14. How is HDFS fault tolerant?
15. Can Name node and Data node be a commodity hardware?
16. Why do we use HDFS for applications having large data sets and not when there are a lot of small files?
17. How do you define "block" in HDFS? What is the default block size in Hadoop 1 and in Hadoop 2? Can it be changed?
18. What does 'jps' command do?
19. How do you define "Rack Awareness" in Hadoop?
20. What is "speculative execution" in Hadoop?
21. How can I restart "Name node" or all the daemons in Hadoop?
22. What is the difference between an "HDFS Block" and an "Input Split"?
23. Name the three modes in which Hadoop can run.
24. What is "MapReduce"? What is the syntax to run a "MapReduce" program?
25. What are the main configuration parameters in a "MapReduce" program?
26. State the reason why we can't perform "aggregation" (addition) in mapper? Why do we need the "reducer" for this?
27. What is the purpose of "RecordReader" in Hadoop?
28. Explain "Distributed Cache" in a "MapReduce Framework".
29. How do "reducers" communicate with each other?
30. What does a "MapReduce Partitioner" do?
31. How will you write a custom partitioner?
32. What is a "Combiner"?
33. What do you know about "SequenceFileInputFormat"?
34. What are the benefits of Apache Pig over MapReduce?
35. What are the different data types in PigLatin?
36. What are the different relational operations in "Pig Latin" you worked with?
37. What is a UDF?
38. What is "SerDe" in "Hive"?
39. Can the default "Hive Metastore" be used by multiple users (processes) at the same time?
40. What is the default location where "Hive" stores table data?